Deploying Virtual Network Functions with Non-uniform Models in Tree-structured Networks

Yang Chen, *Student Member, IEEE,* Jie Wu, *Fellow, IEEE,* and Bo Ji, *Member, IEEE*

***Abstract*—Network Function Virtualization (NFV) has promoted the implementation of network functions from expensive hardwares to software middleboxes. These software middleboxes, also called Virtual Network Functions (VNFs), are executed on switch-connected servers. Efficiently deploying such VNFs on servers is challenging because the traffic rate of flows must be fully processed by their requested VNFs when they reach destinations, and the deployed positions of VNFs are restricted by the server capacity. In addition, each network function offers non-uniform VNF models (types) with different configurations of processing volumes and costs. This paper focuses on minimizing the total cost of deploying VNFs for providing a specific network function to all flows in tree-structured networks. First, we prove the NP-hardness of non-uniform VNF deployment in a tree topology and propose a dynamic programming based solution with a pseudo-polynomial time complexity. Then we narrow it down to three simplified cases by focusing on either uniform VNFs or the linear line topology. Specifically, three algorithms are introduced: an improved dynamic programming based algorithm for deploying uniform VNFs in a tree topology, a performance-guaranteed algorithm for deploying non-uniform VNFs in a linear line topology, and an optimal greedy algorithm for deploying uniform VNFs in a linear line topology. Additionally, we generalize our approach to a case of deploying a service chain, which consists of multiple network functions applied to flows in a specific order. We propose two solutions: one is optimal but time-consuming while another is heuristic but efficient. Extensive simulations are conducted to evaluate our algorithms.**

***Index Terms*—Deployment, NFV, tree-structured networks, VNFs.**

I. Introduction

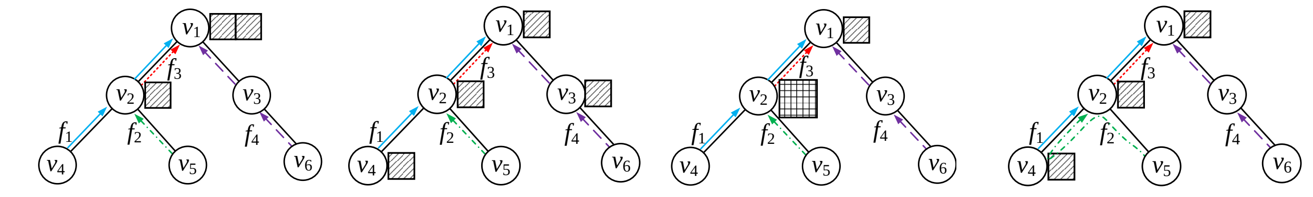
Network Function Virtualization (NFV) addresses the problems of traditional expensive hardwares [1] by leveraging virtualization technology to implement network functions in software modules [2]. These software modules, also called Virtual Network Functions (VNFs) [3], are most commonly provisioned in modern networks to demonstrate their increasing importance [4], such as firewalls, network address translators, proxies, and deep packet inspection. With the emergence of Software Defined Networking (SDN), there is a tendency to incorporate SDN and NFV in concerted ecosystems [5]. SDN allows VNFs to pick service locations from multiple available servers and maneuver traffic through appropriate VNFs; on the other hand, traditional purpose-built hardware appliances leave no choice for allocation [6]. The technical combination of SDN and NFV results in a more flexible architecture and has the potential to reduce capital and operating expenses, shorten product release cycles, and improve service agility [7].

Y. Chen, J. Wu, and J. Bo are with Center for Networked Computing in Temple University, USA. Email: {yang.chen, jiewu, boji}@temple.edu.

In this paper, we study the VNF deployment problem [8] with a given set of flows in tree-structured networks, whose switch-connected servers have limited capacities (the maximum number of deployed VNFs). Tree-structured topologies are quite common in streaming services and Content Delivery Networks (CDNs) [9]. Additionally, it is proven NP-hard to minimize the total number of VNFs to deploy even one single kind of network function with uniform VNFs (each VNF has the same processing volume and setup cost) in a general topology [10]. Thus, we narrow it down to tree-structured networks and provide stronger algorithmic results in this paper. We assume that all flows are upstream (i.e., the destination is closer to the tree root than source) and require to be processed by identical network functions (later extended to an identical service chain, consisting of multiple kinds of network functions serving in a specific order [11]). We initially include non-uniform VNF types for a single kind of network function, which has different configurations of processing volumes and setup costs [12]. The processing volume of a VNF instance (simplified as VNF in the rest of paper) can be shared by multiple flows. A flow can also be fractionally processed by several VNFs before its destination [13]. Our objective is to minimize the total setup cost of deploying VNFs subject to that all flows can be fully processed with their traffic rates when they reach their destinations.

To the best of our knowledge, most existing work on VNF deployment frequently formulate complex IP problems with no efficiency-guaranteed solvers or are limited to design with heuristic solutions. Few proposed solutions can provide provable performance guarantees. Additionally, none of existing work considers the non-uniform VNF types for one function. Here we use an example in Fig. 1 to show the VNF deployment problem that considers non-uniform VNFs and limited vertex capacities. The topology of the toy example is a binary tree with six vertices $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. We are given four flows, f_1, f_2, f_3, and f_4, whose traffic rates are $3, 3, 4$, and 2, respectively. Their paths are given in Figs. 1 (a-c) as the shortest ones, which are $\{v_4, v_2, v_1\}, \{v_5, v_2\}, \{v_2, v_1\}$, and $\{v_6, v_3, v_1\}$, and are arbitrary in Fig. 1 (d). All flows request to be processed with a network function m. There are two types of VNFs for m: a small square VNF with processing volume of 4 and setup cost of 2, and a large square VNF with processing volume of 8 and setup cost of 3. We aim at minimizing the total cost of deploying VNFs subject to that all flows can be fully processed when reaching destinations.

In Fig. 1(a), we are given only one type (uniform) of VNF, the small square one (volume 4 and cost 2). With unlimited vertex capacities, one optimal deployment shown in Fig. 1(a)



(a) Unlimited vertex capacity and uniform VNFs.
(b) Limited vertex capacity and uniform VNFs.
(c) Limited vertex capacity and non-uniform VNFs.
(d) Arbitray routing path with limited vertex capacity and uniform VNFs.

Fig. 1: A motivating example with four flows f_1 (rate 3), f_2(rate 3), f_3(rate 4), f_4(rate 2), small square VNF with volume 4 and cost 2, and large square VNF with volume 8 and cost 3.

can be achieved by applying the algorithm in [10]. All traffic of f_1 and 1 unit of traffic of f_2 are processed by the deployed VNF on v_2, while the rest of the traffic is processed by the two VNFs deployed on v_1. The total setup cost is 6 because three VNFs are deployed. As for the limited vertex capacity case, if each server can place at most one VNF (vertex capacity is one), one optimal deployment with a minimum cost of 8 is shown in Fig. 1(b). Compared to Fig. 1(a), one more VNF is deployed since v_1 can deploy only one VNF. In order to fully process all flows when they reach their destinations, both VNFs on v_1 and v_4 waste 1 processing volume while the one deployed on v_3 wastes 2. This waste is unavoidable in order to satisfy all flows' service requirements of function m; at the same time, it does not violate the vertex capacity constraint as well. In Fig. 1(c), if the vertex capacity is also one, we are given non-uniform types of function m: small (volume 4 and cost 2) and large (volume 8 and cost 3) square VNFs. One optimal deployment shown in Fig. 1(c) has a total cost of 5, which is less than that in Fig. 1(b). The existence of non-uniform VNFs adds more choices for the deployment, which further complicates our problem. When the paths of flows are not given in Fig. 1(d) with the same setting in Fig. 1(b), we can avoid to waste the processing volume of deployed instances by detouring f_2 first to v_4 (through v_2) and then back to its destination v_2. This results in a deployment with a total cost of 6, which is less than that of Fig. 1(b).

The main challenges of our deployment problem lie in the selection of VNF types and locations, as well as the processing volume allocation of each deployed VNF. The limited vertex capacity constraint further complicates the deployment. Flows have to be fully processed before reaching their destinations. Intuitively, if VNFs are deployed too close to the tree root, the processing volume is more likely to be used up; however, flows with destinations far from the root may not be processed because no VNFs are deployed along their paths. If VNFs are deployed too far from the root, the opportunity to share the processing volume of one VNF is scarce, and thus, some volume will be wasted and more VNFs are needed, resulting in a higher total setup cost. Additionally, non-uniform VNFs for a single kind of network function offer more deployment options and make our problem more complex.

In this paper, we initially focus on the deployment of a single kind of network function and include several interesting solutions with provable performance by narrowing down our focus to some special settings. We first solve the non-uniform VNF deployment problem in a tree topology with a dynamic programming based method. Because of the NP-hardness of the problem, the solution is pseudo-polynomial and its time complexity is not tractable. We then study a special case of uniform VNF deployment and improve the dynamic programming solution to have an acceptable time complexity. Additionally, the non-uniform VNF deployment problem in a linear line topology can be transformed to the classic submodular set cover problem so that we introduce a performance-guaranteed greedy strategy. An optimal greedy algorithm is designed for a simple case: uniform VNF deployment in a linear line topology. We then extend our problem more generally to deploy a service chain with limited vertex capacities in a tree-structured network.

Our main contributions are summarized as follows:

- We prove our non-uniform VNF deployment problem is NP-hard in tree-structured networks even when there is only one kind of network function. We also demonstrate that if the routing paths are not determined, our VNF deployment problem can be reduced to the classic Unbounded Knapsack problem, which has been studied with numerous solutions.
- When there is only one kind of network functions, we propose four pseudo-polynomial algorithms in different topology settings and VNF configurations, as shown in Tab. I along with their properties and time complexities[1] ($|V|$: total number of vertices). Since $|M|$ (total number of VNF types of configurations) and $c_{\max}$ (largest vertex capacity) are small and integer-valued, while $w_{\max}$ (largest single VNF setup cost) is in arbitrary precision and order of magnitude, the first algorithm is computationally hard, and the complexities of other three algorithms are dramatically improved.
- We formulate the service chain deployment problem in a tree-structured network with non-uniform VNFs and propose two solutions: one is optimal but time-consuming and another is heuristic but efficient.
- We conduct extensive simulations to evaluate the efficiency of our proposed algorithms.

The remainder of this paper is organized as follows. Section II surveys related work. Section III describes the model, formulates the problem, and shows the hardness of the VNF deployment problem. Section IV introduces our deployment algorithms in tree-structured topologies. In Section V, we handle cases in line topologies. Section VI discusses the service

[1]An algorithm has a pseudo-polynomial time if its running time is a polynomial in the numeric value of the input (the largest integer present in the input) (e.g., $c_{\max}$ in Tab. I), instead of the length of the input (the number of bits required to represent it) (e.g., V in Tab. I), which is the case for polynomial time algorithms.

TABLE I: Our proposed solutions and time complexities.

Topo \ Type	Non-uniform		Uniform	
Tree	DP	Optimal	DP	Optimal
		$O(\lvert V\rvert^4 \times (c_{\max} \times w_{\max})^3)$		$O(\lvert V\rvert^4 \times (c_{\max})^3)$
Line	Greedy	Approximate	Greedy	Optimal
		$O(\lvert V\rvert^2 \times \lvert M\rvert \times c_{\max})$		$O(\lvert V\rvert \times c_{\max})$

chain deployment. Section VII includes the experiments, and Section VIII concludes the paper.

II. Related Work

NFV frameworks have drawn a lot of attention, especially in the area of VNF deployment problems [14, 15]. Various objectives with different backgrounds [16, 17] have been conducted in recent years. In this section, we give a brief review of the state-of-the-art studies.

Casado et al. [18] propose a model for deploying a single type of VNF and present a heuristic algorithm to solve the deployment problem. Seyed et al. [19] tried to solve the resource management problem to determine the number and location of VNFs. However, they only focus on the DDoS attack defense, and the node capacity is only one for each virtual machine. [10] studies the joint deployment and allocation of a single type of VNF, in which flows can be split and fractionally served by several VNFs. They propose several performance-guaranteed algorithms to minimize the number of VNFs. However, they treat all servers with unlimited capacities such that they are able to hold an arbitrary number of VNFs, which is not practical. [20] is the first to study the VNF deployment problems while considering the effects of changing traffic volume. It also studies the multiple VNF deployments of different dependency relationships. It targets load balancing through VNF deployment and flow-routing path selection. However, this work only processes a single flow and does not take consideration of the limited VNF processing volume. This results in exclusive VNFs for each flow, which is wasteful of server resources. Sallam et al. in [21] aim to maximize the total amount of flows while satisfying not only the node capacity constraint but also the resource budget constraint. However, all the above papers only focus on a single kind of VNF and do not take non-uniform VNFs into consideration.

There are other types of service coverage for each flow, such as a service chain where each flow has to be covered by a sequence of services with or without a particular order, instead of a single service coverage used in our model. In [22], Mehraghdam et al. propose a context-free language to formalize the chaining of middleboxes and describe the middlebox resource allocation problem as a mixed integer quadratically constrained program. In [23], Rami et al. locate middleboxes in a way that minimizes both new middlebox setup costs and the distance cost between middleboxes and flows' paths. They provide near optimal approximation algorithms to guarantee a placement with theoretically proven performance. Flowtag [5] architecture uses packet tags that hold the middle-box context. The SDN controller configures switches and middle boxes to use these tags in order to enforce network-wide policies. Both [24] and [25] aim to maximize the number of requests for each service chain. in [24], Kuo et al. propose a systematic way to tune the proper link consumption and the middlebox setup costs in the joint problem of middlebox placement and path selection. [26] aims to minimize the total cost of weighted VNF deployment and flow completion time when flows get processed by a VNF instance in a non-preemptive manner. Allybokus et al. in [27] study the service chain deployment problem with a consideration of the partial order among VNFs and anti-affinity rules. In [28], Richard et al. focus on the latency violation issue on the deployment and migration of VNF instances in a dynamic network situation. [29] models a virtual switching cpu-cost function when a server needs to handles multiple deployed sub-sequences from different service chains and aims at minimizing the total cpu-cost of deployment and switching. However, all these papers do not provide performance-guaranteed solutions and do not take the non-uniform model of a single VNF into consideration, where our main contributions reside. In this paper, being focused on tree-structured networks, we propose optimal DP-based solutions for VNF deployment when there is only one service with non-uniform VNF models. For deploying a service chain with non-uniform VNFs, we also propose two solutions: one is optimal but time-consuming while the other is heuristic but efficient.

III. Model and Formulation

A. Network Model

We first present our model of the directed tree-structured network, $T = (V, E)$, where $V = \{v\}$ is a set of vertices (i.e., switches), and $E = \{e\}$ is a set of directed edges (i.e., links). We use v to denote a single vertex and vertices are labelled as $1, 2, ..., |V|$ by the Breadth-First-Search (BFS). We use $|\cdot|$ to denote the cardinality of a set. Each vertex v_i is connected to a capacity-limited server. The vertex capacity, denoted as c_i, represents the maximum number of VNFs that can be deployed on v_i. For each location on v_i, we can deploy one VNF with any type of configuration. We use two definitions of tree data structure to simplify our discussion.

***Definition** 1 (height, subtree):* The *height* of a vertex is 1 plus the difference between the depth of the tree and the depth of the vertex. A *subtree* T_i of a vertex v_i in a tree T is a tree consisting of v_i and all its descendants in T.

Take the tree in Fig. 1(a) as an example. The height of v_1 is 3 and the heights of v_3 and v_4 are 2 and 1, respectively. The subtree T_2 consists of v_2, v_4, and v_5.

We are given a set of flows $F = \{f\}$, all of which request to be processed by an identical network function (Section 6 extends it to an identical service chain consisting of multiple kinds of network functions applied to flows in a specific order). All flows are upstream flows, i.e. the source of a flow is a descendant of its destination. We make this assumption for ease of presentation only. Our results can be immediately generalized to cases where the flows are either upstream or downstream. We use f to denote a flow with a source of src_f, a destination of dst_f, and an initial traffic rate of r_f. We say that a flow is satisfied when its initial traffic rate is fully processed before reaching its destination.

TABLE II: Symbols and Definitions.

Symbols	Definitions
V, E, F, M	set of vertices, edges, flows, and VNF types
v, f, Ω	a vertex, a flow, and a deployment plan
Ω_v, c_v	the deployment plan and vertex capacity of v
src_f, dst_f, r_f	source, destination, initial traffic rate of f
$m(v,j)$	the j_{th} VNF placed on v
$\alpha(v,j), w(v,j)$	processing volume and setup cost of j_{th} VNF on v
$\lambda^f_{m(v,j)}$	traffic rate of f processed by $m(v,j)$

We define $M = \{m\}$ as the set of VNF types with different configurations for the requested kind of network function. Each VNF type m has a processing volume α_m, which is the maximum total traffic rate that one m VNF can process. There is also a setup cost w_m for setting up one VNF of type m. Different VNF types of configurations for a single kind of network function provide the same network service, but have various processing volumes and setup costs. We simplify the types of different configurations as different types as follows.

***Definition** 2:* [non-uniform, uniform] VNFs are called *non-uniform* if the number of VNF types (for a single kind of network function) is more than one; otherwise, they are called *uniform*.

We assume each flow can be fractionally processed by several VNFs of any type deployed at vertices along its path. We introduce the definitions of the deployment plan and its feasibility as follows.

***Definition** 3 (deployment plan, feasibility):* A *deployment plan of* v, denoted as Ω_v, is a set of VNFs with different types that are deployed on v. These VNFs are labeled by $1, 2, ..., |\Omega_v|$. A *deployment plan of the tree* T, denoted as Ω, is the union set of Ω_v, $\forall v \in V$, i.e. $\Omega = \{\Omega_v | v \in V\}$. We call a deployment plan *feasible* if all flows are fully processed when reaching their destinations.

Note that we can check the existence of a feasible deployment plan by deploying all the VNFs with the maximum processing volume in all available locations of servers. If this deployment plan is still not feasible, then no feasible deployment exists.

We use $m(v,j) \in \Omega_v$ to record the j_{th} VNF placed on v after the labelling. The processing volume and setup cost of $m(v,j)$ are expressed as $\alpha(v,j)$, and $w(v,j)$, respectively. Let $\lambda^f_{m(v,j)}$ denote the amount of f's traffic rate processed by the j_{th} VNF deployed on v. Here each packet of flows should only be processed by VNFs once, because being processed by any VNF will add an extra transmission delay, which should be avoided. In the following, we use the superscript max to denote the maximum value in a set such as $w_{\max} = \max_{m \in M} w_m$ and $c_{\max} = \max_{v \in V} c_v$. For ease of reference, we summarize the notations in Tab. II.

B. Problem Formulation

In this paper, we study the VNF deployment problem: given a set of flows F in a tree-structured network T, we deploy non-uniform VNFs with the minimum total cost to satisfy all requests of flows.

***Definition** 4 (total cost):* The total cost of a deployment plan Ω is the summed-up cost of setting up all VNFs, denoted by $cost(\Omega)$, which satisfies $cost(\Omega) = \sum_{v \in V} \sum_{m(v,j) \in \Omega_v} w(v,j)$.

Our problem can be formulated as:

$$\min_{\Omega} \quad cost(\Omega) \tag{1}$$

$$\text{s.t.} \quad |\Omega_v| \le c_v \qquad \forall v \in V \tag{2}$$

$$\sum_{v \in V} \sum_j \lambda^f_{m(v,j)} \ge r_f \qquad \forall f \in F \tag{3}$$

$$\sum_{f \in F} \lambda^f_{m(v,j)} \le \alpha(v,j) \qquad \forall m(v,j) \in \Omega_v, v \in V \tag{4}$$

Our objective is to minimize the total cost of deployed VNFs in Eq. (1). The decision variables are Ω_v, $\forall v \in V$, which form the deployment plan Ω. Eq. (2) states that the total number of deployed VNFs of each vertex is within its capacity. Eq. (3) guarantees that each flow is fully processed with its initial traffic rate. Eq. (4) requires that the sum of all processed traffic rates of each VNF on each vertex is no more than its processing volume.

C. Problem Hardness Analysis

In a general topology with uniform VNFs, [10] proves that it is NP-hard to minimize the total deployed VNF number, which is equivalent to minimizing the total cost of the deployment. Here we study the hardness of deploying the non-uniform VNFs and we have:

***Theorem** 1:* The *non-uniform VNF deployment* with the minimum cost is NP-hard even to deploy a single kind of network function in a line topology.

Proof: Here we prove our theorem 1. First, we check the feasibility of a deployment plan is in a polynomial time, since we can check in $O(|F|)$ time to make sure that all flows are fully processed when reaching their destinations.

Second, we show that *Unbounded Subset Sum* [30] can be reduced to the *non-uniform VNF deployment*. Consider a case of *Unbounded Subset Sum* with n numbers $W = \{w_1, w_2, ..., w_n\}$ and a target w. To construct a case equivalent to the *non-uniform VNF deployment*, we simplify the deployment problem by having a line topology with unlimited-capacity vertices. We are given a set of flows F, all of whose source and destination are the leftmost and rightmost nodes in the line. Each flow has an initial traffic rate r_f and requests the same network function. We assume the total traffic rate $\sum_{f \in F} r_f$ is equal to the target w of the *Unbounded Subset Sum*, i.e. $\sum_{f \in F} r_f = w$. We are given a set of VNF types M with n types for the requested network function. The setup costs of the VNF types are $w_1, w_2, ..., w_n$, and their processing volumes are the same as the setup costs, meaning $\alpha_i = w_i$. The sum of the processing volumes of the deployed VNFs should be no less than w since all flows needs to be fully processed. When there is no processing volume wasted in a deployment plan, the sum of the processing volumes is exactly w. The total cost of the deployment is $\sum \alpha_j = \sum w_j = w$, which is also the minimum. If we can find such a deployment of VNFs with the costs of $w'_1, w'_2, ..., w'_k$ adding up to the total cost w, then the corresponding numbers in the *Unbounded Subset Sum* VNF can also add up to exactly w.

Conversely, if there are numbers $w'_1, w'_2, ..., w'_k \in W$ adding up to exactly w in the *Unbounded Subset Sum*, then we can deploy the corresponding VNFs with setup costs $w'_1, w'_2, ..., w'_k$;

this is a feasible deployment plan with the minimal total cost w. Consequently, since the *Unbounded Subset Sum* is an NP-complete problem, our *non-uniform VNF deployment* is NP-hard. The theorem holds. ■

It is worth mentioning that we can apply the PTAS solutions in [31] if all flows have the same path, i.e., the topology is a line. However, whether there exists PTAS solutions for the case with general topologies remains an open question. In the following parts, we first focus on deploying a single kind of network function to all flows under different conditions in Sections 4 and 5. In Section 6, we extend to the more general case of deploying a service chain.

Additionally, if the paths of each flow are not given as a priori, we need to route flows beyond finding a deployment plan Ω with the minimum total cost. As the link bandwidth has no constraint in this paper, we tend to use up the processing volumes of deployed VNF instances subject to minimizing the total cost. We can formulate the problem as:

$$\min_{\Omega} \quad cost(\Omega)=\sum_{v\in V}\sum^{j}_{m(v,j)\in\Omega_v} w(v,j) \tag{5}$$

$$s.t. \quad \sum_{v\in V}\sum^{j}_{m(v,j)\in\Omega_v}\alpha(v,j)\geq\sum_{f\in F} r_f \tag{6}$$

$$\text{Eq.(4)} \tag{7}$$

Since there is no flow path limitation, flows can be routed to VNF instances with remaining processing volume. As long as the deployment Ω is feasible, the vertex capacity constraint in Eq. (4) has no influence and can be omitted. The above formulation can be reduced to *Unbounded Knapsack problem* [32]: We multiple both sides of the inequalities 5 and 6 by -1 and do corresponding changes to symbols as:

$$\max_{\Omega} \quad -cost(\Omega)=\sum^{j}_{m(j)\in\Omega} -w(j) \tag{8}$$

$$s.t. \quad \sum^{j}_{m(j)\in\Omega} -\alpha(j)\leq\sum_{f\in F} -r_f \tag{9}$$

$-w(j)$ and $-\alpha(j)$ are the value and the weight of one item. There are several solutions for the classic problem such as dynamic programming and PTAS [32]. After we have found the deployment plan Ω, we can deploy all instances randomly on vertices as long as the total number of deployed instances at any vertex is no more than its capacity. Then we route flows to get processed by deployed instances no matter their paths are longer or have cycles. The insight of the above solution for the case with unknown flow paths is that the influence of network topology is vanished. This is because the routing path of each flow does not affect our objective. However, in practical, a longer routing path will incur a larger transmission delay, which directly influence the network performance. In this paper with a tree topology, we assume all flows are upstream with the shortest path, which complicates the deployment of VNF instances.

IV. VNF Deployment in a Tree Topology

A. *Non-uniform VNF deployment in a tree topology*

First we handle the most general case. We propose a dynamic programming based solution for the non-uniform VNF deployment problem in a tree topology, called the Non-uniform Dynamic Programming algorithm (HeteDP).

Before the recurrence, we define some notations. Let $OPT(i,w)$ denote the minimum total unprocessed rate going out of node v_i by deploying VNFs with a total cost w in the subtree of v_i. If we are unable to fully process flows having $dst_f\in T_i$ by a total cost w, we have $OPT(i,w)=\infty$. This is because the destination is the last chance for a flow to be processed. We prioritize processing flows with smaller-height destinations since their chance of being processed is lower. We use $l(i)$ and $r(i)$ to denote the roots of v_i's left and right subtrees, and $w(l)$ and $w(r)$ to denote the allocated costs of v_i's left and right subtrees, respectively. $Deploy(i,w)$ denotes the maximum total processing volume by deploying VNFs with a total cost w on v_i. The relation of the minimum total unprocessed traffic rates out of v_i and its children can be formulated as:

$$OPT(i,w)=\max\{0,\min_{\substack{w(l)+w(r)\leq w\\ w(l),w(r)\geq 0}}\{\sum_{src_f=v_i} r_f+OPT(l(i),w(l))+ OPT(r(i),w(r))-Deploy(i,w-w(l)-w(r))\}\} \tag{10}$$

Eq. (10) states that $OPT(i,w)$ equals 0 if there is a deployment plan able to process all unprocessed rates by deploying VNFs with a total cost w in the subtree of v_i; otherwise, it equals the minimum total unprocessed traffic rate out of node v_i. We combine all possible allocations of the total cost w among v_i's children and itself by changing $w(l)$ and $w(r)$.

To prove its optimality, let's consider one of the optimal deployments as Ω^* when given a VNF deployment problem. Here are some observations of Ω^*: (i) If Ω^* deploys VNFs with a fixed total cost w in the subtree of a vertex v, it should process as much traffic rate as possible. In other words, the total unprocessed traffic rate going out of v (upwards to its parent) should be minimized with the allocated cost w. This is because the higher the unprocessed traffic rate is when coming out of v, the larger cost the deployment of v's ancestors is likely to have. (ii) The unprocessed traffic rate passing through v comes from two kinds of flows: flows with $src_f=v$ (flows that start at v) and flows with some unprocessed traffic rates and $src_f\in T_v\backslash v$ (not-fully-processed flows coming up from its subtrees). (iii) The total deployment costs of all subtrees of v's children must be no more than w. Suppose each child vertex v_i deploys VNFs with a total cost w_i in the optimal deployment Ω^*, then VNFs with a total cost $w-\sum_{v_i\in T_v} w_i\geq 0$ will be deployed on vertex v. (iv) With a fixed value of w_i for the subtree of v_i, its deployment plan should also have the minimum total unprocessed traffic rate going upward out of v_i in order to lower the potential cost of deployed VNFs of v_i's ancestors. (v) As the optimal deployment should have the minimized unprocessed traffic rate going out of v, the deployed VNFs on v with a total cost $w-\sum_{v_i\in T_v} w_i$ should have the maximum total processing volume.

With the insights above, the objective of our deployment problem is equivalent to finding the minimum cost of making the unprocessed traffic rate out of v_1 as low as 0. Moreover, the optimal deployment of a tree T with the root v_1 is able to be separated into a polynomial number of subproblems in its children. The optimal solutions of its children with different allocated cost combinations yield an optimal deployment to v_1, and we can build up solutions to these subproblems using a recurrence. It is worth mentioning that there are exponential combinations of the costs that are allocated to the vertex itself

Algorithm 1 Non-uniform DP (HeteDP)

```
In: Sets of vertices $V$, edges $E$, flows $F$, VNFs $M$;
Out: The minimum total cost of deployed VNFs and the
     deployment plan $\Omega$;
1: Initiate the array of $OPT$;
2: Generate the array of $Deploy$;
3: for each node $v_i$ from bottom-up do
4:     for $w \in [0, \sum_{v \in T_i} c_v \times w_{\max}]$ do
5:         Use the recurrence Eq. (10) to compute $OPT(i, w)$;
6:         if $OPT(i, w) = 0$ then break;
7: Find $\Omega$ with the minimum $w$ making $OPT(1, w) = 0$;
8: return The deployment plan $\Omega$.
```

and all subtrees of its children when the total cost is fixed. In order to generate the optimal deployment plan, we need to list all such combinations, which is exponential of the number of v's children and w. In this paper, we only discuss the binary tree topology to reduce the number of combinations to polynomials of w. As a result, we can generate an optimal solution with an acceptable time complexity.

As for the item $Deploy(i, w - w(l) - w(r))$ in Eq. (10), we should maximize it in order to minimize the total unprocessed traffic rate out of v_i. This means that the maximum total traffic rate is processed by deploying VNFs on v_i with a cost of $(w - w(l) - w(r))$, which can be formulated as following:

$$\max_{\Omega} \quad \textstyle\sum_{m(i,j) \in \Omega_i} \alpha(i, j) \tag{11}$$

$$s.t. \quad \textstyle\sum_{m(i,j) \in \Omega_i} w_i(j) \le w - w(l) - w(r) \tag{12}$$

$$|\Omega_i| \le c_i \tag{13}$$

The formulation is the same as the classic knapsack problem [33] except for the second constraint. In the knapsack problem, we are given a set of items, each of which has a non-negative weight and a distinct benefit. We need to find a subset with the maximum total benefit subject to the constraints such that the total weight of the subset does not exceed specific values. The processing volume α_m and the setup cost w_m correspond to the benefit and weight in the knapsack problem, respectively. We slightly modify the dynamic programming solution of the knapsack problem proposed in [34]. We use $vol(w)$ to denote the maximum total processing volume that can be attained with a total deployment cost no more than w. The value of $vol(w - w(l) - w(r))$ is the solution to our problem. Suppose $vol(0) = 0$, then the recurrence can be justified as $vol(w) = \max_{m \in M}\{\alpha_m + vol(w - w_m)\}$. When the number of selected items reaches c_v, the total processing volume $vol(w)$ remains unchanged by not adding more items even when the weight w is not used up. This is because we need to control not only the total cost to be less than $w - w(l) - w(r)$, but also ensure that the number of selected items is less than the vertex capacity. Thus, we list all possible deployment combinations on v_i and find the feasible one with the largest processing volume as Ω_v.

Lemma 1: The worst time complexity for generating the $Deploy$ array is $O((c_{\max})^2 \times w_{\max})$.

Proof: The modified knapsack problem can be solved in $O(c_v \times (w - w(l) - w(r)))$ time complexity. We find that the solution to our modified knapsack problem is independent of the deployment plan. In order to lower the time complexity of HeteDP, we can calculate the *Deploy* array in advance and refer to its values when applying the HeteDP algorithm. The worst time complexity of the modified knapsack problem is $O(c_{\max} \times (c_{\max} \times w_{\max})) = O((c_{\max})^2 \times w_{\max})$. This is because the maximum deployment on a vertex is to deploy the most expensive VNF on all available locations. ■

We propose the HeteDP algorithm in Alg. 1. We initiate all values of $OPT(i, w)$ as 0 in line 1. We calculate the recurrence in Eq. (10) for each vertex from bottom-up in lines 3-5. Whenever $OPT(i, w) = 0$, we break the current loop and continue to do the next loop in line 6. We find the minimum w making $OPT(1, w) = 0$ in line 7 and return the corresponding deployment plan Ω by tracing back in line 8. We analyze the time complexity of our algorithm as follows.

Theorem 2: The worst time complexity of HeteDP algorithm is $O(|V|^4 \times (c_{\max} \times w_{\max})^3)$.

Proof: HeteDP algorithm is a pseudo-polynomial time algorithm using a dynamic programming method. First, all $|V|$ vertices need to be traversed so that the algorithm has $|V|$ iterations. Second, in each iteration of a vertex from bottom-up, we try all possibilities of the cost value w. The maximum cost value is $O(\sum_{v \in V} c_v \times w_{\max}) = O(|V| \times c_{\max} \times w_{\max})$. Next, for a fixed cost value w for the subtree of a vertex v, we need to list all combinations of allocating the cost w to itself and its two children while ensuring $w(l) + w(r) \le w$. There are at most $O((|V| \times c_{\max} \times w_{\max})^2)$ combinations. Then for each combination, we need a constant time to calculate the value of $\sum_{src_f = v_i} r_f + OPT(l(i), w(l)) + OPT(r(i) + 1, w(r)) - Deploy(i, w - w(l) - w(r))$ by referring to the OPT array as well as the $Deploy$ array. As discussed in Lemma 1, the generation of all values in the $Deploy$ array takes at most $O((c_{\max})^2 \times w_{\max})$ time and we only need to calculate it once. We determine the minimum value by traversing the values of all combinations in a $O((|V| \times c_{\max} \times w_{\max})^2)$ time and calculate the value of $OPT(i, w)$. Finally, the worst time complexity is the number of iterations, times the maximum number of cost value, times the maximum number of combinations of a fixed cost value, which is $O(|V| \times (|V| \times c_{\max} \times w_{\max}) \times (|V| \times c_{\max} \times w_{\max})^2) = O(|V|^4 \times (c_{\max} \times w_{\max})^3)$. In real systems, the largest server capacity $c_{\max}$ is integer-valued and relatively not too large. The tough case is to handle the cost factor $(w_{\max})^2$ with arbitrary precision and order of magnitudes, which is likely not tractable. ■

Note that in order to improve the efficiency of our algorithm, we can lower its time complexity by stopping increasing w of $OPT(i, w)$ in two cases: the first case is when the smallest w for the vertex v_i appears, meaning $OPT(i, w) = 0$; the second case is when w reaches $\sum_{v \in T_i} c_v \times w_{\max}$. This is because $OPT(i, w) = 0$ when there is no unprocessed traffic rate out of v_i, meaning that VNFs with cost w can process all flows in the subtree of v_i. A larger w is unable to process any more flows, since no unprocessed flow exists. In addition, finding the minimum value of w to make $OPT(1, w) = 0$ is our objective. The second case states the natural upper bound of w that all available locations in the subtree of T_i are deployed by the most expensive VNF.

Theorem 3: HeteDP is optimal for non-uniform VNF deployment in a tree topology.

The detailed proof is omitted due to the optimal property of the dynamic programming method.

B. Uniform VNF deployment in a tree topology

In this subsection, we discuss the uniform VNF, which has the same processing volume and setup cost. First we present a lemma to transform our objective into a simpler equivalent form when there is only one type of VNFs.

Theorem 4: Minimizing the total cost of deployed VNFs with uniform VNFs is equivalent to deploying the minimum number of VNFs.

Proof: As there is only a single type of VNF m, our cost function can be converted to $cost(\Omega) = \sum_{v \in V} |\Omega_v| \times w_m = |\Omega| \times w_m$. Since w_m is a constant, it is the same as minimizing $|\Omega|$, which is the total number of deployed VNFs. ■

Our objective is transformed to minimizing the total number of deployed VNFs when there is only a single type of VNF m. Inspired by HeteDP, we also propose a dynamic programming based algorithm, called HomoDP, which is simpler and more tractable than HeteDP. We replace the total cost w by the total number of deployed VNFs n in each subtree of vertices. We use $OPT(i,n)$ to denote the minimum total unprocessed traffic rate going out of node v_i by deploying n VNFs altogether in the subtree of node v_i. Our target is to find the minimum n, making $OPT(1,n)=0$. If flows with destinations within the subtree of v_i are unable to be fully processed by deploying n VNFs, we have $OPT(i,n)=\infty$. We also prioritize processing flows with smaller-height destinations. We use $l(i)$ and $r(i)$ to denote the roots of v_i's left and right subtrees, and $n(l)$ and $n(r)$ to denote the deployed VNFs in v_i's left and right subtrees, respectively. There are $n-n(l)-n(r)$ VNFs to be deployed on v_i. We replace the $Deploy(i, w-w(l)-w(r))$ by $(n-n(l)-n(r)) \times \alpha_m$. Then we justify the recurrence formula of HomoDP in Eq. (14).

Lemma 2: The relation of the minimum total unprocessed traffic rates out of v_i and its children can be formulated as:

$$OPT(i,n) = \max\{0, \min_{\substack{n(l)+n(r)\le n\\ n(l),n(r)\ge 0}} \{ \sum_{src_f=v_i} r_f + OPT(l(i),n(l)) + OPT(r(i),n(r)) - (n-n(l)-n(r)) \times \alpha_m\}\} \quad (14)$$

Proof: Eq. (14) states that $OPT(i,n)$ equals 0 if there is a deployment plan able to process all traffic rates by deploying n VNFs altogether in the subtree of v_i; otherwise, it equals the minimum total unprocessed traffic rate out of node v_i. We combine all possible allocations of the n VNFs among v_i and its children by changing values of $n(l)$ and $n(r)$. ■

The detailed HomoDP algorithm is proposed in Alg. 2. We initiate all values of $OPT(i,n)$ as 0 in line 1. We calculate the recurrence in Eq. 14 for each vertex from bottom-up in lines 3-5. Whenever $OPT(i,n)=0$, we break the current loop and continue to do the next loop in line 5. We find the smallest n that makes $OPT(1,n)=0$ in line 6 and return the corresponding deployment plan Ω by tracing back in line 7.

To better understand Alg. 2, we use the topology in Fig. 1(b) with the same settings as an example to show the deployment procedure. The tree has six nodes with capacities $c_v=1, \forall v \in V$. There are four flows f_1, f_2, f_3, and f_4 with initial traffic rates $r_1=3, r_2=3, r_3=4$, and $r_4=2$. There is only one type of VNF m with $\alpha_m=4$. We aim to find the smallest n such that $OPT(1,n)=0$. For ease of reference, we list the values of $OPT(i,j)$ in Table III. We traverse vertices from bottom-up by first calculating $OPT(5,0)=r_2-0=4$. We have $OPT(5,1)=\max\{0, r_2-1\times\alpha_m\}=\max\{0,3-4\}=0$. As $c_5=1$, more than one VNF is unable to be deployed resulting in $OPT(5,n)=0$, $\forall n \geq 2$. Similarly, we can calculate $OPT(3,n)$ and $OPT(4,n)$, $\forall 0 \leq n \leq 4$. Since f_2 with $dst_2=v_2$ is not processed by not deploying any VNF in the subtree of v_2 ($n=0$), we have $OPT(2,0)=\infty$, indicating the infeasibility of the deployment. The detailed calculation of $OPT(2,1)$ is that $OPT(2,1)=\max\{0, \min\{r_3+OPT(4,1)+OPT(5,0)-0\times\alpha_m, r_3+OPT(4,0)+OPT(5,1)-0\times\alpha_m, r_3+OPT(4,0)+OPT(5,0)-1\times\alpha_m\}\}=\max\{0,\min\{4+3+0-0, 4+3+0-0, 4+3+3-4\}=6$. Similarly, we calculate other values of OPT array in Tab. III. The smallest n making $OPT(1,n)=0$ is 4. By tracing back the table, the optimal deployment Ω is as shown in Fig. 1(b).

Note that we can also lower the time complexity of Alg. HomoDP by stopping increasing n in two cases: the first one is when the smallest n (which is 4) for node v_i appears making $OPT(i,n)=0$; the second one is when $n \geq \sum_{v \in T_i} c_v$. The reasons are similar to the explanation for Alg. HeteDP.

Theorem 5: The worst time complexity of the HomoDP algorithm is $O(|V|^4 \times (c_{\max})^3)$.

Proof: Here we analyze the time complexity of the worst case. All the $|V|$ nodes need to be traversed so that the algorithm has $|V|$ iterations. In each iteration of a node v_i from bottom-up, we try all possibilities of the value n. The largest n is $\sum_{v \in V} c_v = O(|V| \times c_{\max})$. For a fixed value n, we need to list all combinations of allocating n VNFs to v_i and all subtrees of its children by changing the values of $n(l)$ and $n(r)$. When $n(l)=0$, $n(r)$ can be $0,1,...,n$; and when $n(l)=n$, $n(r)$ could only be 0. We have $(n+1)+n+...+1=(n+1)(n+2)/2=O(n^2)$ combinations, which is at most $O((|V| \times c_{\max})^2)$. For each combination, we only need a constant time to calculate the value of $\sum_{src_f=v_i} r_f + OPT(2i,n(l)) + OPT(2i+1,n(r)) - (n-n(l)-n(r)) \times \alpha_m$ in the recurrence of Eq. 14 by referring to the OPT array. Next we determine the minimum value in at most $O(|V| \times c_{\max})$ time by traversing values of all combinations and calculating the value of $OPT(i,n)$. Thus, the worst time complexity is the

Algorithm 2 Uniform DP (HomoDP)

In: Middlebox m and sets of vertices V, edges E, flows F;
Out: The minimum number of deployed middleboxes and the deployment plan Ω;

```
1: Initiate the array of $OPT$;
2: for each node $v_i$ from bottom-up do
3:     for $n = 0, 1, ..., \sum_{v \in T_i} c_v$ do
4:         Use the recurrence Eq. 14 to compute $OPT(i,n)$;
5:         if $OPT(i,n) = 0$ then break;
6: Find $\Omega$ with the minimum $n$ that $OPT(1,n) = 0$;
7: return The deployment plan $\Omega$.
```

TABLE III: The values of $OPT(i, n)$ of Fig. 1(b).

i \ n	0	1	2	3	4
1	∞	∞	∞	∞	0
2	∞	6	2	0	0
3	2	0	0	0	0
4	3	0	0	0	0
5	3	0	0	0	0
6	2	0	0	0	0

number of iterations, times the maximum number of n, times the maximum number of combinations of a fixed n, which is $O(|V| \times (|V| \times c_{\max}) \times (|V| \times c_{\max})^2) = O(|V|^4 \times (c_{\max})^3)$, which is a polynomial of $|V|$. Because of the term $c_{\max}$, it is qualified as a pseudo-polynomial time algorithm. Since n is integer-valued, and $\sum_{v \in V} c_v$ is not too large in a real system, the time complexity of HomoDP algorithm is acceptable. ■

***Theorem** 6:* HomoDP is optimal for uniform VNF deployment in a tree topology.

The detailed proof is also omitted due to the optimal property of the dynamic programming method.

V. VNF Deployment in a Line Topology

In this section, we simplify the tree-structured topologies into lines in order to generate more efficient algorithms.

A. Non-uniform VNF deployment in a line topology

In this subsection, we simplify the tree topology into a line and propose a performance-guaranteed algorithm of deploying non-uniform VNFs. We are given a line topology $L = (V, E)$ with $|V|$ nodes (vertices), which are labelled $1, 2, ..., |V|$ by a line coordinate axis. For simplicity, we say that one vertex is *smaller (or larger)* than another vertex if its coordinate is smaller (or larger) and vice versa. Assume the source of each flow is smaller than its destination no matter where its source and destination reside in the line. This means that flows transfer from left to right. When deploying one new VNF of type m on v, we omit the sequence number of the j_{th} VNF $m(v, j)$ by denoting the VNF as $m(v)$. The new deployment plan is expressed as $\Omega \cup m(v)$. Before proposing our solution, we introduce two definitions.

***Definition** 5 (benefit function):* The benefit function, denoted as $b(\Omega)$, indicates the total processed traffic rate of a deployment plan Ω, which satisfies $b(\Omega) = \sum_{v \in V} \sum_{m(v,j) \in \Omega_v} \sum_{f \in F} \lambda^f_{m(v,j)}$.

***Definition** 6 (marginal benefit):* The marginal benefit, denoted as $b_\Omega(m(v)) = b(\Omega \cup m(v)) - b(\Omega)$, indicates the marginal contribution of processing flows by deploying a new VNF of type m on v beyond the current deployment Ω.

We analyze the property of the benefit function $b(\Omega)$. A function f is *submodular* if and only if $\forall S \subseteq T \subseteq N, \forall e \in N \setminus T$, $f_T(e) \leq f_S(e)$. Then we prove that $\forall m(v) \notin \Omega'$, if $\Omega \subseteq \Omega'$, the submodular property holds, i.e., $b(\Omega \cup m(v)) - b(\Omega) \geq b(\Omega' \cup m(v)) - b(\Omega')$.

***Theorem** 7:* $b(\Omega)$ is a submodular function.

Proof: $b(\Omega)$ is a non-decreasing function, which is monotone. Suppose we have two deployments Ω and Ω' with $\Omega \subseteq \Omega'$. It is intuitive that the more VNFs are selected, the fewer unprocessed traffic rates remain, since the newly added m can only process the unprocessed rate. The maximum marginal benefit of a VNF m is α_m because of its processing volume limitation. If the newly added VNF processes no traffic rate in both Ω and Ω', then $b(\Omega \cup m(v)) - b(\Omega) = b(\Omega' \cup m(v)) - b(\Omega') = 0$. As long as m processes some flows in Ω', it will process no less traffic rate in Ω. We have $b(\Omega \cup m(v)) - b(\Omega) \geq b(\Omega' \cup m(v)) - b(\Omega')$. Thus, $b(\Omega)$ is a submodular function. ■

Algorithm 3 Non-uniform VNF deployment in Line

In: Sets of vertices V, edges E, flows F and VNFs M;
Out: The deployment plan Ω;

```
1: $\Omega = \emptyset$;
2: while not all flows are fully processed do
3:     Select $m(v)$ with $\min_{m \in M, v \in V} cost(m(v))/b_\Omega(m(v))$ to handle superior flows;
4:     $\Omega = \Omega + m(v)$;
5: return The deployment plan $\Omega$.
```

Here we explain that our problem formulation in Section III(B) can be transformed to the classic submodular set cover problem [35]. Our objective $cost(\Omega)$ in Eq. (1) is a non-decreasing function. The two constraints in Eqs. (2) and (4) are included in the definition of our $b(\Omega)$ function. Specifically, the ground set of the benefit function $b(\Omega)$ limits the available deploying locations within each vertex's capacity, and the marginal benefit limits the largest contribution of one VNF no more than its processing volume. $b(\Omega)$ is the non-decreasing, submodular set function proved in Theorem 7. The constraint in Eq. (3) corresponds to the covering requirement of the set cover problem that each flow needs to be fully processed. Then our problem can be transformed as:

$$\min_\Omega \qquad cost(\Omega) \tag{15}$$

$$s.t. \qquad b(\Omega) \geq \textstyle\sum_{f \in F} r_f \tag{16}$$

Before introducing the solution, we sort flows in alphabetical order of a tuple $< dst_f, src_f >$ (the ascending order of destination and the descending order of source). We include two new definitions.

***Definition** 7:* (*prior, superior*) A flow f is *prior* to a flow f' if: (1) $dst_f < dst_{f'}$; (2) $dst_f = dst_{f'}$ and $src_f > src_{f'}$. A flow f is *superior* if no flow is prior to f.

The priority of flows indicates their order of achieving the processing volume of one VNF, and superior flows should be processed first because of their small destinations or shorter path lengths. We propose a greedy algorithm in Alg. 3, called Non-uniform VNF deployment in Line algorithm (HVPL) to solve the deployment problem. We initiate the deployment plan as an empty set in line 1. In lines 2-3, we iteratively select $m(v)$ with the minimum value of $w_m/b_\Omega(m(v))$ to handle superior flows. Then we add the deployment of the new VNF to the current plan Ω until all flows are fully served. The deployment plan Ω returns in line 4.

***Theorem** 8:* The worst time complexity of HVPL algorithm is $O(|V|^2 \times |M| \times c_{\max})$.

Proof: In each round, we have at most $|V|$ vertices and $|M|$ types of VNFs. The maximum number of rounds is to

TABLE IV: The values of $cost(m)/b_{\Omega}(m(v))$ of Fig. 2.

Ω \ $m(v)$	$m(1)$	$m(2)$	$m(3)$	$m(4)$	$m(5)$	$m(6)$	$m'(1)$	$m'(2)$	$m'(3)$	$m'(4)$	$m'(5)$	$m'(6)$	$m''(1)$	$m''(2)$	$m''(3)$	$m''(4)$	$m''(5)$	$m''(6)$
$\emptyset$	\	2	2	2	2	2	3	1.5	1.5	1.5	1.5	3	4	1	1	1	2	2
$\{m''(2)\}$	2	∞	2	2	2	2	3	∞	1.5	1.5	1.5	1.5	4	∞	1	1	2	2
$\{m''(2), m''(3)\}$	∞	∞	∞	2	2	2	∞	∞	∞	3	3	3	∞	∞	∞	4	4	4

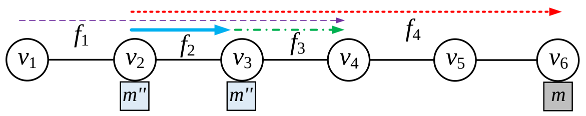


Fig. 2: Illustration of the HVPL algorithm, where $r_1 = 1, r_2 = 4, r_3 = 2, r_4 = 2$, $\alpha = 1, \alpha' = 2$, $\alpha'' = 4$, and $w = 2, w' = 3, w'' = 4$.

place VNFs in every available location in servers, which is $\sum_{v \in V} c_v = O(c_{\max} \times |V|)$. Thus, the worst time complexity of HVPL is the maximum number of rounds times the choices in each round, which is $O(|V| \times |M| \times (c_{\max} \times |V|) = O(|V|^2 \times |M| \times c_{\max})$. ■

To better understand Alg. 3, we use an example shown in Fig. 2 to illustrate the deployment procedure. In this example, the line topology has six vertices with $c_v = 1, \forall v \in V$. We are given a set of non-uniform VNFs, $M = \{m, m', m''\}$. Their processing capacities are $\alpha = 1, \alpha' = 2$, and $\alpha'' = 4$, and setup costs are $w = 2, w' = 3$, and $w'' = 4$, respectively. There are four flows f_1, f_2, f_3, and f_4, whose paths are shown in Fig. 2 and initial traffic rates are $r_1 = 1, r_2 = 4, r_3 = 2$, and $r_4 = 2$, respectively. The alphabetical order of flows is $f_2 > f_3 > f_1 > f_4$. For each round, we calculate $w_m/b_{\Omega}(m(v))$, $\forall v \in V, m \in M$. The algorithm is then conducted as follows: (1) we list all possible deployments over the current empty deployment plan $\Omega = \emptyset$ in the second row of Tab. IV. The smallest one is $w''/b(m''(2)) = 1$. As a result, we deploy an m'' VNF on v_2. We prioritize processing the superior flowf_2. (2) referring to the third row of Tab. IV, the smallest one is $w''/b_{\{m''(2)\}}(m''(3)) = 1$. As a result, we deploy an m' VNF on v_3 to process f_1 and f_4. (3) referring to the fourth row of Tab. IV, the smallest is $w/b_{\{m''(2), m''(3)\}}(m(6)) = 2$. Thus, we deploy an m VNF on v_6 and so all flows are satisfied. We return the feasible deployment plan $\Omega = \{m''(2), m''(3), m(6)\}$.

Theorem 9: The proposed Alg. 3, HVPL, can achieve a deployment with at most $H(\max_{m(v)} b_{\emptyset}(m(v)))$ times of the minimum cost, where $H(d) = \sum_{i=1}^{d} \frac{1}{i}$.

Proof: Our VNF deployment problem has the same formulation as the submodular set cover [35] and the chosen deployment plan Ω exactly corresponds to its greedy algorithm in Section 2 in [35]. Hence, the approximation ratio follows from Theorem 1 in [35]. $\max_{m(v)} b_{\emptyset}(m(v))$ is the maximum benefit of only deploying a specific VNF m. ($\emptyset$: empty set) ■

B. Uniform VNF deployment in a line topology

In this subsection, we discuss the uniform VNF deployment in a line topology. First we present a lemma to transform our objective into a simpler equivalent form based on Theorem 4.

Theorem 10: Minimizing the total cost of deployed VNFs with uniform VNFs is also equivalent to minimizing the total amount of wasted processing volume.

Algorithm 4 Greedy VNF Placement (GVP)

In: VNF m and sets of vertices V, edges E, flows F;
Out: the deployment plan Ω;

```
1:  Sort flows in the alphabetical order;
2:  for each vertex $v$ from 1 to $|V|$ do
3:      while the sum of unprocessed traffic rate of superior flows passing $v$ is no less than $\alpha_m$ and $c_v > 0$ do
4:          Allocate one new VNF on $v$;
5:      if the superior flow $f$ with $dst_f \leq v$ have some unprocessed traffic rate then
6:          if $c_{v'} \leq 0,\ \forall src_f \leq v' \leq v$ then
7:              return Non-existence of a feasible plan;
8:          else
9:              Allocate one VNF on $\max v', \forall c_{v'} > 0, v' < v$;
10:     Reallocate the processing volumes of all deployed VNFs from left to right for flows in alphabetical order;
11: return The deployment plan $\Omega$.
```

Proof: From Theorem 4, $|\Omega|$ is minimized. Because $\sum_{f \in F} r_f$ is a fixed value and α_m is also a constant, $|\Omega| \times \alpha_m - \sum_{f \in F} r_f$, which is the total wasted processing volume of deployed VNFs, is also minimized. ■

Here we further simplify the settings by deploying a uniform VNF in a line topology. We propose a greedy algorithm called Greedy VNF Plan (GVP), and prove its optimality for minimizing the deployment cost. The algorithm is shown in Alg. 4. The insight of GVP is to minimize the total processing volume waste based on Theorem 10 when all flows are satisfied. Superior flows are the first to be processed, and GVP only deploys VNFs when no processing volume is wasted or the superior flow reaches its destination. In GVP, we sort flows in an alphabetical order in line 1. In lines 2-10, we traverse vertices from left to right. When the vertex v has remaining capacities and the total unprocessed traffic rate of superior flows passing v can use up a new VNF's processing volume α_m in line 3, we deploy one new VNF on v in line 4. In lines 5-9, we handle the case where superior flows cannot use up the processing volume of a new VNF. We reallocate the processing volumes of deployed VNFs in line 10 while the deployment plan Ω is returned in line 11.

Theorem 11: The worst time complexity of GVP algorithm is $O(|V| \times c_{\max})$.

Proof: We deploy VNFs at $|V|$ vertices, and for each vertex v, we place at most c_v VNFs. In each loop we place at least one VNF in a constant time. The maximum number of loops is $\sum_{v \in V} c_v = O(|V| \times c_{\max})$. Thus, the worst time complexity of Alg. 4 is the maximum number of loops $O(|V| \times c_{\max})$. ■

For a better understanding, we use an example shown in Fig. 3 to illustrate the deployment procedure. Each vertex has a capacity of 1, i.e. $c_v = 1,\ \forall v \in V$. There are four flows

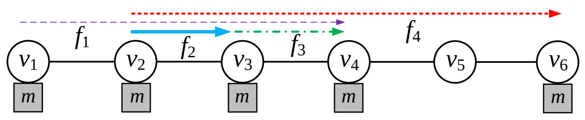


Fig. 3: Illustration of the GVP algorithm, where $r_1 = 1, r_2 = 4, r_3 = 2, r_4 = 2$, and $\alpha_m = 2$.

f_1, f_2, f_3, and f_4 whose initial traffic rates are $r_1 = 1, r_2 = 4, r_3 = 1$, and $r_4 = 1$, respectively. There is only one type of VNF m with a processing volume $\alpha_m = 2$. The alphabetical order of flows is $f_2 > f_3 > f_1 > f_4$. First, we deploy one VNF on v_2 because f_2 is the superior flow and its unprocessed traffic rate is larger than α_m. The same happens to v_3 so we also deploy one new VNF to process the remaining traffic rate of f_2. Then f_2 is satisfied and f_3 becomes the superior flow. We place one new VNF on v_4 because f_3's unprocessed traffic rate is larger than α_m. After that, f_1 is the superior flow. Since v_2, v_3, and v_4 have no remaining capacities and v_1 is the largest vertex with $c_1 > 0$ along the path of f_1, one VNF has to be deployed on v_1 even when 1 processing volume is wasted; otherwise, f_1 cannot be fully processed when reaching its destination v_4. Afterwards, f_4 becomes the superior flow and we deploy one VNF on v_6. All flows are satisfied and the final deployment plan is $\{v_1, v_2, v_3, v_4, v_6\}$ shown in Fig. 3.

***Lemma** 3:* By applying Alg. 4, a VNF at v has remaining volume only when: suppose f has the lowest priority among all satisfied flows, then no flow f' with $src_{f'} \leq v$ and $dst_{f'} \geq dst_f$ has any unprocessed traffic rate. All flows prior to f' certainly use up the capacity from next vertex of v to dst_f.

Proof: Alg. 4 deploys a new VNF on a vertex v only when: (1) the unprocessed traffic rate of superior flows passing v is larger than the processing volume of a VNF; (2) one flow f has some unprocessed rate and there is no capacity left from the next vertex of v to dst_f. The first situation has no processing volume waste. In the second situation, the last VNF with the remaining processing volume has to be deployed; otherwise, the flow f cannot be satisfied before reaching its destination since no vertex capacity is available from v to dst_f. ■

***Theorem** 12:* Alg. 4 is optimal for deploying the uniform VNF in a line topology.

Proof: We prove the optimality of Alg. 4 by induction. In Theorem 10, we demonstrate that the objective is equivalent to minimizing the total waste of deployed VNFs. We list all situations where VNFs have remaining processing volumes. Suppose v_1 is the smallest vertex with such one VNF, then all the other VNFs deployed on and before v_1 have no remaining volume. From Lemma 3, the VNF has to be deployed and no more unprocessed traffic rate goes right from the vertex v_1. Thus, there is no deployment that has the waste less than Ω. Assume it is true for all vertices less than v_k, which indicates that no more superior unprocessed traffic rate can go right from the vertex v_k. Then the situation of the next vertex, having one VNF with some remaining volume, is the same as the situation of the first vertex v. This is because there is no unprocessed traffic rate of a flow f with $src_f \leq v_k$, and so we are able to treat the next vertex of v_k as the new origin. Repeatedly, we find the smallest $v > v_k$ with one VNF having the remaining capacity. Additionally, we have proven that it is true for the smallest v. So it is also true for the v_{k+1} with one VNF having the remaining capacity. To sum up, Alg. 4 has the least amount of wasted volume, which is equivalent to deploying the least number of VNFs. ■

VI. Service Chain Deployment in a Tree Topology

In this section, we extend our model to study a more general case: the service chain deployment in a tree topology. First, we mathematically formulate the service chain deployment problem based on Section 3.2. Next, we propose one optimal but time-consuming solution and one heuristic but efficient solution, indicating the tradeoff between the performance and the time efficiency.

Before proposing our solutions, we need to introduce some notations. We are given a service chain $\mathcal{M} = \{M_i\}$ with multiple network functions, each of which consists of several VNF types, i.e., $M_i = \{m\}$. We use the subscript i to indicate the i_{th} required function in the chain, meaning that the chain has a serving order of $M_1 \to M_2 \to \ldots \to M_{|\mathcal{M}|}$. Each flow needs to be processed by the same service chain $\mathcal{M}$. We define $t(v, j)$ as the type of the j_{th} placed VNF on node v. Our problem can be formulated as:

$$\min_{\Omega} \quad cost(\Omega) \tag{17}$$

$$\text{s.t.} \quad |\Omega_v| \leq c_v \qquad \forall v \in V \tag{18}$$

$$\sum_{f\in F} \lambda^f_{m(v,j)} \leq \alpha(v,j) \qquad \forall m(v,j) \in \Omega_v, v \in V \tag{19}$$

$$\sum_{\substack{v'\in T_v \\ t(v',j)=M_p}} \lambda^f_{m(v',j)} \geq \sum_{\substack{v'\in T_v \\ t(v',j)=M_q}} \lambda^f_{m(v',j)} \quad \text{if } p<q, \ \forall v \in V \tag{20}$$

$$\sum_{v\in V} \sum_{j:t(v,j)=M_{|\mathcal{M}|}} \lambda^f_{m(v,j)} \geq r_f \qquad \forall f \in F \tag{21}$$

Our objective in Eq. (17) is the same as in Eq. (1): minimizing the total cost of deployed VNFs. The decision variables are also Ω_v, $\forall v \in V$, which form the deployment plan Ω. Eq. (18) states that the total number of deployed VNFs of each vertex is within its capacity. Eq. (19) requires that the sum of processed traffic rate by each VNF is no more than its processing volume on all vertices. Eq. (20) ensures the serving order constraint for the service chain. It indicates that if function M_p must be served before M_q in the service chain ($p < q \leq |\mathcal{M}|$), then for a flow f along its path from its source src_f to the current node v, all processed traffic rates of f by M_p should be no less than all its already-processed traffic rates by M_q. Note that any node v' along the path of an upstream flow f from its source src_f to the current node v resides in the subtree of v, and thus, we use the expression of $v' \in T_v$. Eq. (21) guarantees that each flow $f \in F$ is fully processed with its initial traffic rate r_f by the last function of its required service chain.

The formulation is an integer program problem. Its optimal solution can be found by using the brute force method to list all deployment plans. We solve the problem by applying the Integer Programming Solver Cplex [21]. For simplicity, we call this result OPT. However, we find that when the network scale is large, the execution time is too long, which shows that it is time consuming. This is not acceptable in real systems even though it can find the optimal solution. As a result, we also propose an efficient heuristic algorithm called Sevice Chain Deployment in Tree (SCDT), as shown in Alg. 5.

Algorithm 5 Service Chain Deployment in Tree (SCDT)

In: Sets of vertices V, edges E, flows F and VNFs M;
Out: The deployment plan Ω;

```
1: $\Omega = \emptyset$;
2: for each function $M_i \in \mathcal{M}$ in the serving order do
3:     while not all flows are fully processed do
4:         Select $m(v)$ with $\min_{m \in M_i} cost(m(v))/b_{\Omega}(m(v))$ to handle superior flows (feasibility check);
5:         If multiple $m(v)$ have the same minimum value, $v$ closer to leaf nodes is preferred;
6:         $\Omega = \Omega + m(v)$;
7: return The deployment plan $\Omega$.
```

Alg. SCDT is based on Alg. HVPL, proposed in Section 5.1. Line 1 initiates the deployment plan as an empty set since none of the VNFs are deployed. In lines 2-6, we select each network function $M_i \in \mathcal{M}$ in the service chain one by one, strictly following their serving order. For each network function M_i, Alg. SCDT extends from Alg. HVPL to deploy non-uniform types of VNFs. We select the type of VNF and its deployed position (vertex) with the minimum value of $w_m/b_{\Omega}(m(v))$ to handle superior flows. Note that we need to do the feasibility check because the traffic must be processed by the previous VNFs in the chain so that the current VNF can process it. In order to reduce the infeasible situation, we prefer the vertex closer to leaf vertices when several $m(v)$ have the same smallest value. Line 7 returns the deployment plan Ω.

VII. Evaluation

A. Settings

Topology: We test the impact of the topology scale with a fixed flow number of 1000 and basic settings as follows, and the results are shown in Fig. 4. All their total costs have little variance with the vertex number increment. Thus, we only conduct our simulations in a line topology and a random-generated tree topology, both of which empirically have fixed 20 vertices. Each switch vertex is connected to a server with an identical capacity of 10, i.e. $c_v = 10$, $\forall v \in V$. Additionally, traditional data center networks and WAN design over-provision the network with $30-40\%$ of the average network utilization in order to handle traffic demand changes and failures [36]. As a result, 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 the concern in this paper.

VNFs: We conduct the simulations with two sets of VNFs, M and M'. The first set M only includes one type of VNF m, i.e. $M = \{m\}$. Its required server resource is 2, i.e. $w_m = 2$. The processing volume of one m VNF is 8, i.e., $\alpha_m = 8$. The second set M' includes three types of VNFs, i.e. $M = \{m, m', m''\}$. Their processing volumes are $\alpha = 6, \alpha' = 8$, and $\alpha'' = 10$, and costs are $w = 1, w' = 2$ and $w'' = 3$.

Traffic: All flows' paths are fixed and their traffic rates are also known a priori. Under the tree topology, the source of each flow is a descendant of its destination. We adopt the flow

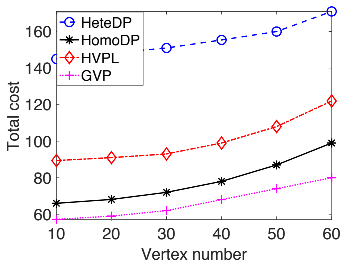


Fig. 4: The impact of topology scale.

size distribution of Facebook data centers, which is collected in 10-minute packet traces of three different node types: a Web-server rack, a single cache follower, and a Hadoop node [37]. More than 88% flows are less than 7 Mbps. As a result, the traffic rate ranges from 0.1 to 6 Mbps with a granularity of 0.1 Mbps and is generated randomly in this paper.

B. Comparison algorithm and performance metrics

We include two benchmark schemes in our simulations:

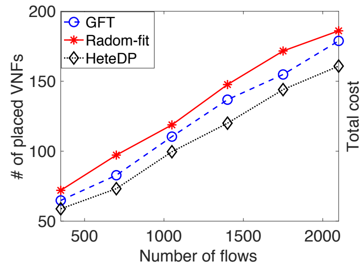
- Sang et al. [10] propose the algorithm GFT for deploying only one type of VNF without the constraint of vertex capacity. VNFs are not deployed until the destinations of some flows need to be served.
- *Random-fit* randomly deploys non-uniform VNFs on random nodes on the paths until all flows are fully served.

GFT is only designed for deploying the uniform VNFs. When we need to deploy non-uniform VNFs, we randomly select a single type of VNF each time and apply GFT to deploy it. Additionally, if the vertex capacity is not enough, we simply deploy the VNFs in its nearest descendants with enough remaining capacities until all flows are fully served.

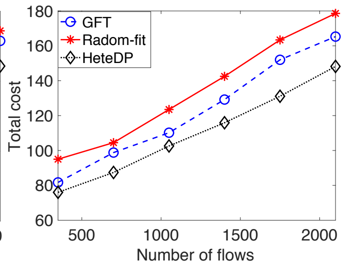
We use three performance metrics: the total number of deployed VNFs, the total cost (non-uniform VNFs), and the average server utilization (uniform VNF) for benchmark comparisons. The total number of deployed VNFs is the sum of deployed VNFs of each type. We also evaluate the total cost corresponding to our objective function as shown in Eq. (1). Since all vertex capacity settings are identical, the average required server volume is equivalent to the total consumed server volume divided by the total number of servers.

C. Results of the VNF deployment in a tree topology

Fig. 5 shows the results of the non-uniform VNF deployment in a tree topology. We have tested the algorithms with 350 to 2010 flows. All their sources and destinations are randomly generated. As for the total number of VNFs, HeteDP deploys the fewest VNFs and significantly outperforms the other two as shown in Fig. 5(a). The numbers of deployed VNFs by the three methods are approximately 3 times the numbers when we only need to deploy a single type of VNF. In Fig. 5(b), HeteDP has the smallest average server utilization ratio. When there are 2100 flows, HeteDP uses 19.8% less of the total cost than Random-fit and 17.8% less than GFT. This is because HeteDP checks all possible deployment cases and selects the optimal one with the minimum cost. Note that the execution time of HeteDP is ten times that of GFT and Random-fit because of DP's optimality.

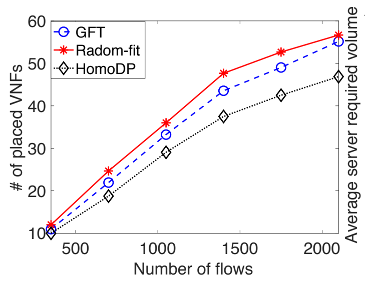


(a) Total number of deployed VNFs.

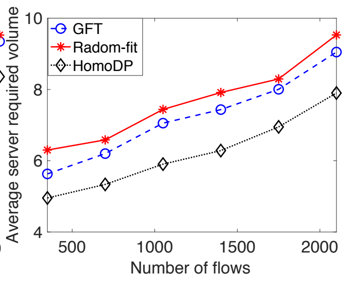


(b) Total cost.

Fig. 5: Non-uniform VNF deployment in a tree topology.



(a) Total number of deployed VNFs.



(b) Avg. server required volume.

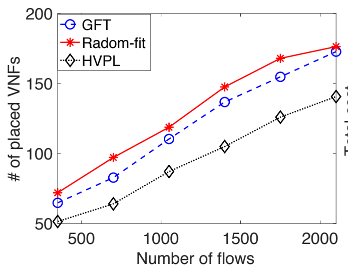
Fig. 6: Uniform VNF deployment in a tree topology.

Fig. 6 is the result of uniform VNF deployment in a tree topology. We use the same flow set as the one in Fig. 5. The results are shown in Fig. 6(a) and Fig. 6(b). The number of deployed VNFs by HomoDP ranges from 11 to 53, which is always much smaller than the other two. When there are 2100 flows, HomoDP deploys 18.5% less VNFs than Random-fit and 16.7% less than GFT. The gap among these three methods becomes larger as more flows involved in the network. We also notice that GFT has a much more similar performance to Random-fit in the general topology. This is because that GFT is designed for the tree topology and requires no constraint on vertex capacity. In terms of the average server utilization, HomoDP is at least 17.1% less than the other two no matter how many flows are generated because HomoDP considers the allocation of the vertex capacity resources.

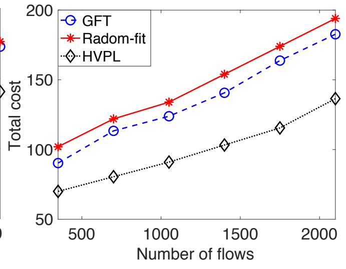
D. Results of the VNF deployment in a line topolgy

Fig. 7 shows the results of the non-uniform VNF deployment in a line topology. Alg. HVPL also performs better than GFT and Random-fit. We have tested the algorithms with 350 to 2100 flows. The advantage of our algorithm becomes sharper when there are more flows in the network. This is because it is less likely to waste the spare processing volumes in the deployed VNFs. With more flows, the traffic load is so heavy that the total cost increases significantly. This illustrates that the capacities in all servers are almost used up and more processing volumes of deployed VNFs are wasted. When there are 2100 flows, the total cost of our HVPL algorithm is 32.0% less than Random-fit.

Results of the uniform VNF deployment in a line topology are shown in Fig. 8(a) and Fig. 8(b). In Fig. 8(a), the numbers of deployed VNFs by the three methods are approximately one third of the numbers when we only need to deploy non-uniform VNFs. As the capacity in the server is relatively sufficient, the increasing tendencies of the results are gentle.

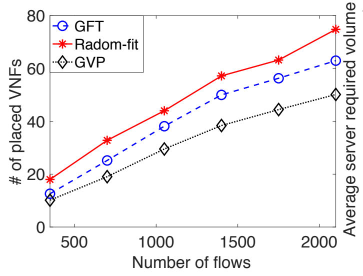


(a) Total number of deployed VNFs.

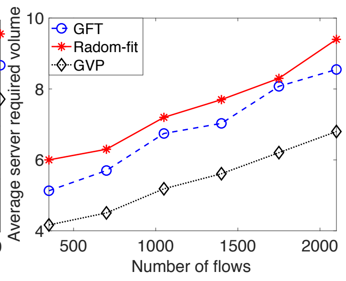


(b) Total cost.

Fig. 7: Non-uniform VNF deployment in a line topology.



(a) Total number of deployed VNFs.



(b) Avg. server required volume.

Fig. 8: Uniform VNF deployment in a line topology.

Our GVP method has the best performance both in the number of deployed VNFs and the average server utilization. The difference is more obvious when the number of flows is larger. This is because GVP is optimal for deploying a single type of VNFs with the constraint of vertex capacity while the other two are not. When there are 2100 flows, GVP deploys 21.6% fewer VNFs than Random-fit and 14.4% fewer than GFT. In terms of server utilization, GVP always has the lowest ratio.

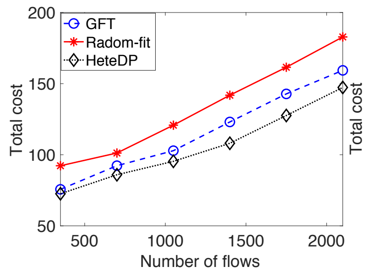
E. Results with a larger vertex capacity

To evaluate the impacts of vertex capacity, we enlarge each vertex's capacity from 10 to 20, i.e., $c_v = 20, \forall v \in V$, and other settings remain unchanged. Due to space limitation, we only list the results of the total cost in all four cases of topologies and VNF types of configurations. The basic tendencies of all curves are similar to the results with $c_v = 10, \forall v \in V$. Our algorithms and GFT improve their performances with a smaller total cost. It's worth mentioning that the difference between GFT and each of our algorithms is reduced. This is because a larger vertex capacity is closer to the case without the vertex capacity constraint, where GFT is the optimal solution. However, we find that Random-fit performs even a little worse because there are more available locations.

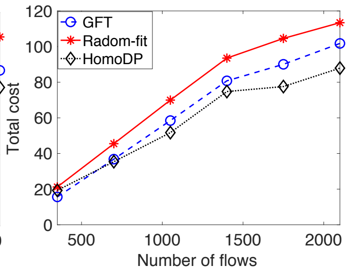
F. Results of the service chain deployment

Fig. 10 shows the results of the service chain deployment in a tree topology. We apply the same topology setting. The service chain has a length of 6 and each network service has 3 types of VNFs. We enlarge the server capacity to 20, i.e. $c_v = 20,\ \forall v \in V$. Additionally, we evaluate an extra metric: the execution time of algorithms.

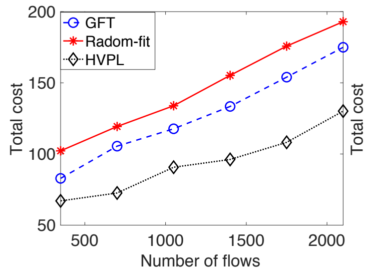
Fig. 10(a) evaluates the execution times of the four algorithms. Alg. OPT runs for the longest time while it always has the best performances on other metrics in Figs. 10(b)-(d). It is worth mentioning that our Alg. SCDT has a longer execution time than the other two. This is because it has the selection



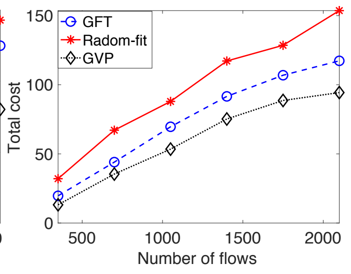
(a) Non-uniform VNFs in a tree.



(b) Uniform VNFs in a tree.



(c) Non-uniform VNFs in a line.



(d) Uniform VNFs in a line.

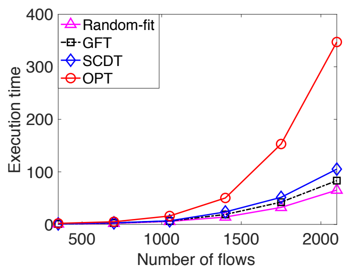
Fig. 9: Total cost.

procedure among all nodes, which requires us to calculate all values. In Fig. 10(b), Alg. OPT has the lowest number of deployed VNFs and Alg. SCDT has the second lowest. It indicates the high performance of our proposed algorithms. Fig. 10(c) shows the results of comparing the objectives of our paper. On average, the total cost of Alg. SCDT is only 14.6% more than that of Alg. OPT while the other two have at least 29.5% higher cost than Alg. OPT. The result of the average server required volume is shown in Fig. 10(d). The difference between Alg. SCDT and Alg. GFT is not obvious since both of them are based on selecting local minimum values to deploy VNFs. Alg. Random-fit always requires the highest volume because the random selection wastes a lot of the processing volumes of deployed VNFs, which in turn requests more VNFs to be deployed in order to meet all flows' requirements.

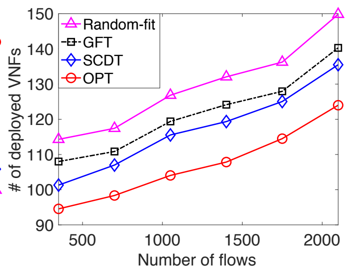
In summary, the simulations verify the correctness and efficiency of our proposed algorithms in the tree and line topologies. They also show that only considering a single type of VNF deployment is too one-sided, because all types of VNFs need to share the limited server resources. It is worth mentioning that our HVPL and GVP can be used as efficient, greedy algorithms with significant insights in all kinds of tree topologies and traffic distributions. Additionally, the general topologies can also be transformed to the combination of several trees by grouping flows and then applying our algorithms. The results also verify the necessity of the vertex constraint.

VIII. Conclusion

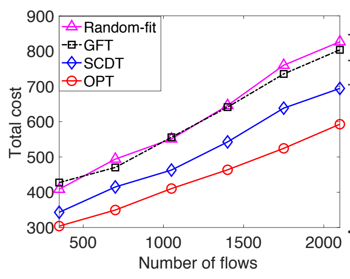
We study the joint VNF deployment and flow allocation problem. We aim to minimize the total cost of deploying VNFs when all flows are fully processed. Initially, we assume that all flows request one same network function. We study the non-uniform VNF deployment in tree topologies. First, we prove the NP-hardness of the deployment and propose a DP solution. Then we introduce an improved DP solution for uniform VNFs in a tree topology. We reformulate the deployment of non-uniform VNFs in a line and propose a performance-guaranteed



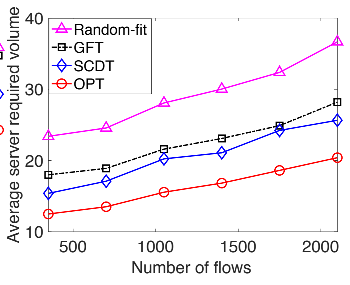
(a) Execution time.



(b) Total number of deployed VNFs.



(c) Total cost.



(d) Average server required volume.

Fig. 10: Service chain deployment.

strategy. An optimal greedy solution is designed for uniform VNF deployment in a line. Then, we study a more general model in which all flows request a service chain, consisting of multiple network functions serving in a specific order. Extended trace-driven simulations prove the correctness and efficiency of our algorithms.

It is worth mentioning that the vertex capacity constraint in terms of the maximum number of VNFs can be extended to a constraint on the total resource capacity. Setting up each type of VNF needs different amounts of the vertex resource besides different setup costs. Hence, our DP solution, HeteDP, needs to include one more dimension of the available resource in the current vertex. In this case, the *Deploy* item in the DP formulation becomes a 2-D knapsack problem. Although the extension can still be addressed in a DP formulation, we leave detailed treatment to our future work.

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] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *SIGCOMM 2012*.

[2] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, "NFVnice: Dynamic backpressure and scheduling for NFV service chains," in *SIGCOMM 2017*.

[3] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *SIGCOMM 2014*.

[4] J. Sherry, S. Ratnasamy, and J. S. At, "A survey of enterprise middlebox deployments," in *Semantic Scholar*, 2012.

[5] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *NSDI 2014*.

[6] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "Nfp: Enabling network function parallelism in nfv," in *SIGCOMM 2017*.

[7] Y. Li and M. Chen, “Software-defined network function virtualization: A survey,” *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[8] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, “Network slicing and softwarization: A survey on principles, enabling technologies, and solutions,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[9] S. Seyyedi and B. Akbari, “Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes,” in *CNDS 2011*.

[10] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, “Provably efficient algorithms for joint placement and allocation of virtual network functions,” in *INFOCOM 2017*.

[11] Y. Chen and J. Wu, “Nfv middlebox placement with balanced set-up cost and bandwidth consumption,” in *ICPP 2018*.

[12] P. Duan, Q. Li, Y. Jiang, and S. T. Xia, “Toward latency-aware dynamic middlebox scheduling,” in *ICCCN 2015*.

[13] D. Li, P. Hong, K. Xue, and j. Pei, “Virtual network function placement considering resource optimization and sfc requests in cloud datacenter,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664–1677, 2018.

[14] L. Qu, C. Assi, and K. Shaban, “Delay-aware scheduling and resource optimization with network function virtualization,” *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016.

[15] J. Liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, “Incremental server deployment for scalable nfv-enabled networks,” in *INFOCOM 2020*.

[16] A. Laghrissi and T. Taleb, “A survey on the placement of virtual resources and virtual network functions,” *IEEE Communications Surveys & Tutorials 2018*.

[17] G. Mirjalily and L. Zhiquan, “Optimal network function virtualization and service function chaining: A survey,” *Chinese Journal of Electronics*, vol. 27, no. 4, pp. 704–717, 2018.

[18] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, “Virtualizing the network forwarding plane,” in *PRESTO 2010*.

[19] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, “Bohatei: Flexible and elastic ddos defense,” in {*USENIX*} *Security 2015*.

[20] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, “Traffic aware placement of interdependent nfv middleboxes,” in *INFOCOM 2017*.

[21] G. Sallam and B. Ji, “Joint placement and allocation of virtual network functions with budget and capacity constraints,” *arXiv preprint arXiv:1901.03931*, 2019.

[22] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *CloudNet 2014*.

[23] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, “Near optimal placement of virtual network functions,” in *INFOCOM 2015*.

[24] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, “Deploying chains of virtual network functions: On the relation between link and server usage,” in *INFOCOM 2016*.

[25] Y. Li, L. T. X. Phan, and B. T. Loo, “Network functions virtualization with soft real-time guarantees,” in *INFOCOM 2016*.

[26] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, “Design and evaluation of algorithms for mapping and scheduling of virtual network functions,” in *NetSoft 2015*.

[27] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, “Virtual function placement for service chaining with partial orders and anti-affinity rules,” *Networks*, vol. 71, no. 2, pp. 97–106, 2018.

[28] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, “Dynamic, latency-optimal vnf placement at the network edge,” in *INFOCOM 2018*.

[29] M. C. Luizelli, D. Raz, and Y. Sa’ar, “Optimizing NFV chain deployment through minimizing the cost of virtual switching,” in *INFOCOM 2018*.

[30] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[31] S. Martello, “Knapsack problems: algorithms and computer implementations,” *Wiley-Interscience series in discrete mathematics and optimization*, 1990.

[32] K. Jansen and S. E. Kraft, “A faster fptas for the unbounded knapsack problem,” *European Journal of Combinatorics*, vol. 68, pp. 148–174, 2018.

[33] G. B. Mathews, “On the partition of numbers,” *Proceedings of the London Mathematical Society*, vol. s1-28, no. 1, pp. 486–490, 1896.

[34] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[35] L. A. Wolsey, “An analysis of the greedy algorithm for the submodular set covering problem,” *Combinatorica*, vol. 2, no. 4, pp. 385–393, Dec 1982.

[36] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, “B4: Experience with a globally-deployed software defined wan,” in *SIGCOMM 2013*.

[37] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, “Inside the social network’s (datacenter) network,” in *SIGCOMM 2015*.

Yang Chen received her B.Eng. degree in Electronic Engineering and Information Science from University of Science and Technology of China in 2015. She is currently a Ph.D. candidate in the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, USA. Her current research focuses on Software Defined Networks, especially resource allocation, flow scheduling and network updates.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, Journal of Parallel and Distributed Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a Fellow of the AAAS and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Bo Ji (S’11-M’12-SM’18) received his B.E. and M.E. degrees in Information Science and Electronic Engineering from Zhejiang University, Hangzhou, China, in 2004 and 2006, respectively, and his Ph.D. degree in Electrical and Computer Engineering from The Ohio State University, Columbus, OH, USA, in 2012. Dr. Ji joined Department of Computer and Information Sciences (CIS) at Temple University in July 2014, where he is currently an assistant professor. He is also a faculty member of the Center for Networked Computing (CNC) at Temple University. Prior to joining Temple University, he was a Senior Member of the Technical Staff with AT&T Labs, San Ramon, CA, from January 2013 to June 2014. His research interests are in the modeling, analysis, control, optimization, and learning of computer and networking systems, such as communication networks, information-update systems, cloud/datacenter networks, and cyber-physical systems. Dr. Ji is a senior member of the IEEE and a member of the ACM. He is a National Science Foundation (NSF) CAREER awardee (2017) and an NSF CISE Research Initiation Initiative (CRII) awardee (2017).