Joint Placement and Allocation of Virtual Network Functions with Budget and Capacity Constraints

Gamal Sallam and Bo Ji
Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA
Email: {tug43066, boji}@temple.edu

***Abstract*—With the advent of Network Function Virtualization (NFV), network services that traditionally run on proprietary dedicated hardware can now be realized using Virtual Network Functions (VNFs) that are hosted on general-purpose commodity hardware. This new network paradigm offers a great flexibility to Internet service providers (ISPs) for efficiently operating their networks (collecting network statistics, enforcing management policies, etc.). However, introducing NFV requires an investment to deploy VNFs at certain network nodes (called VNF-nodes), which has to account for practical constraints such as the deployment budget and the VNF-node capacity. To that end, it is important to design a joint VNF-nodes placement and capacity allocation algorithm that can maximize the total amount of network flows that are fully processed by the VNF-nodes while respecting such practical constraints. In contrast to most prior work that often neglects either the budget constraint or the capacity constraint, we explicitly consider both of them. We prove that accounting for these constraints introduces several new challenges. Specifically, we prove that the studied problem is not only NP-hard but also non-submodular. To address these challenges, we introduce a novel relaxation method such that the objective function of the relaxed placement subproblem becomes submodular. Leveraging this useful submodular property, we propose two algorithms that achieve an approximation ratio of $\frac{1}{2}(1-1/e)$ and $\frac{1}{3}(1-1/e)$ for the original non-relaxed problem, respectively. Finally, we corroborate the effectiveness of the proposed algorithms through extensive evaluations using both trace-driven simulations and simulations based on synthesized network settings.**

I. Introduction

The advent of Network Function Virtualization (NFV) has made it easier for Internet service providers (ISPs) to employ various types of functionalities in their networks. NFV requires the replacement of network services that traditionally run on proprietary dedicated hardware with software modules, called Virtual Network Functions (VNFs), which run on general-purpose commodity hardware [1]. A wide variety of network functions (firewalls, intrusion detection systems, WAN optimizers, etc.) can be applied to flows passing through network nodes that host VNFs (called VNF-nodes). A flow must be fully processed at one or multiple VNF-nodes so that the potential benefits introduced by NFV can be harnessed [2]. The new network paradigm enabled by NFV not only offers a great flexibility of introducing new network functions, but it also reduces capital and operational expenditure. Therefore, major ISPs have already started the process of transforming their technologies and operations to support NFV [3].

This work was supported in part by the NSF under Grant CNS-1651947.

However, such moves often take place in multiple stages due to the budget limit; in each stage, only a subset of nodes can be selected for deploying/placing VNFs. Moreover, VNF instances typically have a limited capacity, which is shared for processing multiple passing flows. Therefore, given a deployment budget and capacity limit, it is of critical importance to choose a best subset of nodes to become VNF-nodes and to determine the optimal capacity allocation so as to maximize the amount of network traffic passing through them.

In contrast to most prior work that often neglects either the budget constraint (e.g., [4], [5]) or the capacity constraint (e.g., [2]), we explicitly consider both constraints and *formulate a joint problem of VNF-nodes placement and capacity allocation (VPCA)*. The VPCA problem has two main components: VNF-nodes placement and VNF-nodes capacity allocation, which are tightly coupled with each other. That is, deciding where to place the VNF-nodes depends on how the capacity of the VNF-nodes will be allocated; determining an optimal capacity allocation apparently depends on where the VNF-nodes are placed. The challenges posed by this problem are two folds. First, *the placement and capacity allocation subproblems are both NP-hard*. Second, even if we assume that there is an oracle that can optimally solve the capacity allocation subproblem, *the placement subproblem is non-submodular*. This is in stark contrast to the previously studied problem without the capacity constraint [2], which has been shown to be submodular and can be approximately solved using efficient greedy algorithms.

To that end, we introduce a *novel relaxation method* that allows us to design efficient algorithms with constant approximation ratios for the studied VPCA problem. We summarize our key contributions as follows.

- First, we formulate the VPCA problem with budget and capacity constraints as an Integer Linear Program (ILP). Then, we provide an in-depth discussion about the new challenges introduced by the budget and capacity constraints. Specifically, we show that the placement and capacity allocation subproblems are both NP-hard. Further, we show that the objective function of the placement subproblem is not submodular.
- To address these challenges, we relax the requirement of fully processed flows and allow partially processed flows to be counted. This simple relaxation enables us to prove that the relaxed placement subproblem is submodular based on

a *novel network flow reformulation* of the relaxed capacity allocation subproblem. Leveraging this useful submodular property, we design two efficient algorithms that achieve an approximation ratio of $\frac{1}{2}(1 - 1/e)$ and $\frac{1}{3}(1 - 1/e)$ for the original (non-relaxed) VPCA problem, respectively. To the best of our knowledge, *this is the first work that exploits this type of relaxation method to solve a non-submodular optimization problem with provable performance guarantees.*

- Finally, we evaluate the performance of the proposed algorithms using both trace-driven simulations and simulations based on synthesized network settings. The simulation results show that the proposed algorithms perform very closely to the optimal solution obtained from an ILP solver and better than another algorithm [6].

The rest of the paper is organized as follows. First, we position our work compared to related work in Section II. Next, we describe the system model and problem formulation in Section III and discuss the challenges of the VPCA problem in Section IV. Then, we introduce the VPCA relaxation and reformulation in Section V and the proposed algorithms in Section VI. Finally, we present the numerical results in Section VII and conclude the paper in Section VIII.

Due to space limitations, most of the proofs are omitted and provided in our online technical report [7].

II. Related Work

There has been a large body of work that studies the placement problem in different contexts such as NFV, SDN, and edge cloud computing. In NFV, a placement is usually considered at a scale of VNF instances, i.e., where and how many instances of each network function should be placed and allocated [4], [8], [9], [10]. Different objectives are considered in each of them. The problem of how to meet the demand from all of the flows with a minimum cost (e.g., in terms of the number of instantiated instances) is considered in [4], [11]. An extension of these work considers the setting where each flow must traverse a chain of network functions instead of just one function [12]. A similar problem is also considered in [8], [13] but for an online setting where flows arrive and leave in an online fashion. In [10], they consider a joint problem of VNF placement and routing, aiming to minimize the total consumed resources, while a similar problem is considered in [9], with a different objective of ensuring network stability. Also, in [14], the authors consider the placement of a minimum number of nodes to achieve the original maximum flow under a given service function chaining constraint.

In [15], the authors consider the problem of placement and scheduling in the edge clouds. They show that the problem is not submodular in general. Then, they develop a heuristic for the general problem and also identify a special case where the problem becomes submodular, which can be solved efficiently. In [2], the authors consider the selection of a set of nodes to upgrade to SDN. By assuming that the SDN nodes have an infinite capacity, they show that the problem is submodular. Similarly, the work in [5] considers the placement of middleboxes to keep the shortest paths between all communicating pairs under a certain threshold. They show that the problem without a budget constraint is submodular. Different from these studies, we consider both limited VNF-node processing capacity and limited budget constraints. It is more realistic to account for both constraints, which actually brings new challenges that will be discussed in Section IV.

III. System Model and Problem Formulation

We consider a network graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, with $V = |\mathcal{V}|$, and $\mathcal{E}$ is the set of edges connecting nodes in G. We have a set of flows $\mathcal{F}$, with $F = |\mathcal{F}|$. We use λ_f to denote the traffic rate of flow $f \in \mathcal{F}$. The traffic of flow f will be sent along a predetermined path (e.g., a shortest path), and the set of nodes along this path is denoted by $\mathcal{V}_f$. We use $\mathcal{F}_\mathcal{U}$ to denote the set of all flows whose path has one or more nodes in a given set $\mathcal{U}$, i.e., $\mathcal{F}_\mathcal{U} = \{f \in \mathcal{F} \mid \mathcal{V}_f \cap \mathcal{U} \neq \emptyset\}$. When a node is able to support some VNFs, we call it a VNF-node. Since ISPs have a limited budget to deploy VNFs in their networks, they can only choose a subset of nodes $\mathcal{U} \subseteq \mathcal{V}$ to become VNF-nodes. The traffic rate λ_f of each flow can be split and can be processed at multiple VNF-nodes. We use λ_f^v to denote the portion of flow f that is assigned to VNF-node v and use $\boldsymbol{\lambda} \in \mathcal{R}^{F \times V}$ to denote the assignment matrix.

As we mentioned earlier, the benefits of processed traffic can be harnessed from fully processed flows, i.e., flows that have all of their traffic processed at VNF-nodes. Hence, when a flow traverses VNF-nodes and there is a sufficient capacity on these VNF-nodes to process all of its rate, i.e., $\sum_{v \in \mathcal{V}_f \cap \mathcal{U}} \lambda_f^v \geq \lambda_f$, then the flow is counted as a processed flow. Therefore, the total processed traffic can be expressed as follows:

$$J_1(\mathcal{U}, \boldsymbol{\lambda}) \triangleq \sum_{f \in \mathcal{F}} \lambda_f \mathbf{1}_{\{\sum_{v \in \mathcal{V}_f \cap \mathcal{U}} \lambda_f^v \geq \lambda_f\}}, \tag{1}$$

where $\mathbf{1}_{\{.\}}$ is the indicator function. Note that each VNF-node v has a limited processing capacity, denoted by c_v. Hence, the total traffic rate assigned to a node should satisfy the following capacity constraint:

$$\begin{cases} \sum_{f \in \mathcal{F}} \lambda_f^v \leq c_v, & \forall v \in \mathcal{U}, \\ \lambda_f^v = 0, & \forall f \in \mathcal{F} \text{ and } \forall v \notin \mathcal{U}. \end{cases} \tag{2}$$

We assume that the largest traffic rate of any flow is no larger than the smallest processing capacity of any node[1]. Also, we consider a limited budget, denoted by B, and require that the total cost of introducing VNF-nodes do not exceed B. We use b_v to denote the cost of making node v a VNF-node. Hence, the total cost of VNF-nodes should satisfy the following budget constraint:

$$\sum_{v \in \mathcal{U}} b_v \leq B. \tag{3}$$

The above budget constraint limits the number of nodes that can become VNF-nodes, and we may only have a subset

[1]While some studies (e.g., [4]) consider the placement of VNF instances and allow the flow rate to be larger than the capacity of a VNF instance, we consider the problem of placing VNF-nodes, each of which can host multiple VNF instances. Therefore, it is reasonable to assume that the capacity of such a VNF-node is larger than the rate of any flow.

of flows that traverse some VNF-nodes. Accounting for the above deployment budget and VNF capacity constraints, we consider a joint problem of VNF-nodes placement and capacity allocation (VPCA). The objective is to choose a best subset of nodes to become VNF-nodes and optimally allocate their capacities so as to maximize the total amount of fully processed traffic. We provide the mathematical formulation of the VPCA problem in the following:

$$\begin{aligned} \underset{\mathcal{U}\subseteq\mathcal{V},\boldsymbol{\lambda}}{\text{maximize}} \quad & J_1(\mathcal{U},\boldsymbol{\lambda}) \\ \text{subject to} \quad & (2),(3). \end{aligned} \tag{$P1$}$$

IV. Challenges of VPCA

Here, we will identify the unique challenges of the VPCA problem formulated in $(P1)$. We first decompose the VPCA problem into two subproblems: 1) placement: how to select a subset of nodes to become VNF-nodes and 2) allocation: for a given set of VNF-nodes, how to divide their capacity for processing a subset of flows. Then, we prove that both subproblems are NP-hard and that the placement subproblem is non-submodular. This is very different from similar problems neglecting the capacity constraint (2) [2], which have been shown to be submodular and can be approximately solved.

A. NP-hardness

First, we present the formulations of the two subproblems. We start with the allocation subproblem because it will be used in the placement subproblem. For a given set of VNF-nodes $\mathcal{U}\subseteq\mathcal{V}$, let $J_2^{\mathcal{U}}(\boldsymbol{\lambda})$ denote the total amount of fully processed traffic under flow assignment $\boldsymbol{\lambda}$. Note that $J_2^{\mathcal{U}}(\boldsymbol{\lambda})$ has the same expression as that of $J_1(\mathcal{U},\boldsymbol{\lambda})$ in Eq. (1). The superscript $\mathcal{U}$ of $J_2^{\mathcal{U}}(\boldsymbol{\lambda})$ is to indicate that it is associated with a given set of VNF-nodes $\mathcal{U}$. Then, the capacity allocation subproblem for a given set of VNF-nodes $\mathcal{U}$ can be formulated as

$$\underset{\boldsymbol{\lambda}:(2)\ \text{is satisfied}}{\text{maximize}} \quad J_2^{\mathcal{U}}(\boldsymbol{\lambda}). \tag{$P2$}$$

Let $J_3(\mathcal{U}) \triangleq \max_{\boldsymbol{\lambda}:(2)\ \text{is satisfied}} J_2^{\mathcal{U}}(\boldsymbol{\lambda})$ denote the optimal value of problem $(P2)$ for a given set of VNF-nodes $\mathcal{U}$. Then, the placement subproblem can be formulated as

$$\begin{aligned} \underset{\mathcal{U}\in\mathcal{V}}{\text{maximize}} \quad & J_3(\mathcal{U}) \\ \text{subject to} \quad & (3). \end{aligned} \tag{$P3$}$$

Note that in order to solve problem $(P3)$, we need to solve problem $(P2)$ to find the optimal $\boldsymbol{\lambda}$ for a given set of VNF-nodes $\mathcal{U}$. In the following theorem, we will show that both subproblems $(P2)$ and $(P3)$ are NP-hard.

Theorem 1. *The capacity allocation subproblem* $(P2)$ *and the placement subproblem* $(P3)$ *are both NP-hard.*

B. Non-submodularity

Note that the objective function $J_3(\mathcal{U})$ of the placement subproblem $(P3)$ is a set function. At first glance, problem $(P3)$ looks like a submodular maximization problem, which has been extensively studied in the literature and can be approximately solved using efficient algorithms [16], [17]. However,

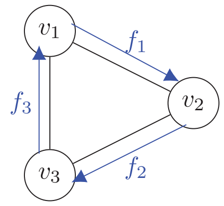


Fig. 1: An example to show non-submodularity of $J_3(\mathcal{U})$

we will show that the objective function $J_3(\mathcal{U})$ is generally non-submodular, which makes the placement subproblem $(P3)$ and the overall problem $(P1)$ much more challenging. We first give the definition of submodular functions.

Definition 1. *For a finite set of elements $\mathcal{V}$, a function H: $2^{\mathcal{V}}\rightarrow\mathbb{R}$ is submodular if for any subset $\mathcal{V}_1\subseteq\mathcal{V}_2\subseteq\mathcal{V}$ and any element $v\in\mathcal{V}\backslash\mathcal{V}_2$, we have*

$$H(\mathcal{V}_1\cup\{v\})-H(\mathcal{V}_1)\geq H(\mathcal{V}_2\cup\{v\})-H(\mathcal{V}_2). \tag{4}$$

The above definition exhibits an important property of diminishing returns. In our problem, if the VNF-node capacity is infinite, i.e., there is no capacity constraint (2), then a flow f can always be fully processed as long as its path has at least one VNF-node, i.e., $\mathcal{V}_f\cap\mathcal{U}\neq\emptyset$. In this case, the total processed traffic $J_1(\mathcal{U},\boldsymbol{\lambda})$ can be rewritten as

$$J_1'(\mathcal{U})=\sum_{f\in\mathcal{F}}\lambda_f\mathbf{1}_{\{\mathcal{V}_f\cap\mathcal{U}\neq\emptyset\}}, \tag{5}$$

where the capacity allocation becomes irrelevant as it does not impact the value of function $J_1'(\mathcal{U})$. It has been shown in [2] that the function $J_1'(\mathcal{U})$ is monotonically nondecreasing and submodular. In this special case, problem $(P1)$ with objective function $J_1'(\mathcal{U})$ can be approximately solved using efficient greedy algorithms.

However, using the example presented in Fig. 1, we show that the objective function $J_3(\mathcal{U})$ is no longer submodular if the VNF-nodes have a limited capacity. Consider three flows: flow f_1 with path $v_1\rightarrow v_2$, flow f_2 with path $v_2\rightarrow v_3$, and flow f_3 with path $v_3\rightarrow v_1$. Assume that each VNF-node has a capacity of 3, and each flow has a traffic rate of 2. If node v_3 is the only VNF-node, then it can only support one flow because its capacity is 3. Therefore, the marginal contribution of adding node v_3 as a VNF-node to the empty set is $J_3(\{v_3\})-J_3(\emptyset)=2-0=2$. Now, assume that before making node v_3 a VNF-node, node v_2 is already a VNF-node, which can support one flow. By making node v_3 a VNF-node, all three flows can be fully processed, and hence, the total processed traffic becomes 6, i.e., the marginal contribution of adding node v_3 to the set $\{v_2\}$ is $J_3(\{v_2\}\cup\{v_3\})-J_3(\{v_2\})=6-2=4>J_3(\{v_3\})-J_3(\emptyset)=2$. This violates the definition of submodular set functions in Eq. (4).

In [18], a notion called supermodular degree is introduced to characterize the level of violation of submodularity for a set function. For problems with a bounded supermodular degree, the authors of [18] propose a greedy algorithm with performance guarantees for the considered problem with a non-submodular objective function. However, the proposed greedy algorithm has two main limitations. First, its approximation ratio is a function of the supermodular degree, which, in our case, could be as large as the number of nodes in the network. Second, its complexity is exponential in the supermodular degree and could be prohibitively high when the supermodular degree is large.

Therefore, our problem $(P1)$ is much more challenging than other similar problems studied in prior work, where the objective function is submodular or has a bounded supermodular degree. To that end, in the next section we will address the aforementioned unique challenges by introducing a novel relaxation and a problem reformulation, which enable us to propose two algorithms with constant approximation ratios.

V. Relaxation and Reformulation

In this section, we present a relaxation of the VPCA problem that allows partially processed flows to be counted. Further, we introduce a novel network flow reformulation of the relaxed capacity allocation subproblem. Both of these techniques will be utilized in designing two efficient approximation algorithms in the next section.

A. Relaxed VPCA Formulation

We first introduce the relaxed VPCA problem, which allows partially processed flows to be counted. In the relaxed VPCA problem, any fraction of flow f processed by VNF-nodes in $\mathcal{V}_f \cap \mathcal{U}$ will be counted in the total processed traffic. That is, the relaxed $J_1(\mathcal{U}, \boldsymbol{\lambda})$ can be expressed as follows:

$$R_1(\mathcal{U}, \boldsymbol{\lambda}) \triangleq \sum_{f \in \mathcal{F}} \sum_{v \in \mathcal{V}_f \cap \mathcal{U}} \lambda_f^v. \tag{6}$$

Apparently, the total processed traffic of flow f cannot exceed λ_f, i.e., the following constraint needs to be satisfied:

$$\sum_{v \in \mathcal{U}} \lambda_f^v \leq \lambda_f, \quad \forall f \in \mathcal{F}. \tag{7}$$

Then, the relaxed version of problem $(P1)$ becomes

$$\begin{aligned} \underset{\mathcal{U} \subseteq \mathcal{V}, \boldsymbol{\lambda}}{\text{maximize}} \quad & R_1(\mathcal{U}, \boldsymbol{\lambda}) \\ \text{subject to} \quad & (2), (3), (7). \end{aligned} \tag{Q1}$$

Next, we decompose problem $(Q1)$, in the same way as we did for problem $(P1)$, into placement and allocation subproblems. For a given set of VNF-nodes $\mathcal{U} \subseteq \mathcal{V}$, let $\Lambda^{\mathcal{U}}$ be the set of all flow assignment matrices λ that satisfy the capacity constraint (2) and the flow rate constraint (7), and let $R_2^{\mathcal{U}}(\boldsymbol{\lambda})$ be the total processed traffic, which has the same expression as that of $R_1(\mathcal{U}, \boldsymbol{\lambda})$ but has $\mathcal{U}$ in the superscript so as to indicate that this function is for a given set of VNF-nodes $\mathcal{U}$. Then, the capacity allocation subproblem for a given set of VNF-nodes $\mathcal{U}$ can be formulated as

$$\underset{\boldsymbol{\lambda} \in \Lambda^{\mathcal{U}}}{\text{maximize}} \quad R_2^{\mathcal{U}}(\boldsymbol{\lambda}). \tag{Q2}$$

Now, let $R_3(\mathcal{U}) \triangleq \max_{\boldsymbol{\lambda} \in \Lambda^{\mathcal{U}}} R_2^{\mathcal{U}}(\boldsymbol{\lambda})$ denote the optimal value of problem $(Q2)$ for a given set of VNF-nodes $\mathcal{U}$. Then, the placement subproblem can be formulated as

$$\begin{aligned} \underset{\mathcal{U} \subseteq \mathcal{V}}{\text{maximize}} \quad & R_3(\mathcal{U}) \\ \text{subject to} \quad & (3). \end{aligned} \tag{Q3}$$

Note that although the relaxed placement subproblem $(Q3)$ can still be shown to be NP-hard, we will prove that the objective function $R_3(\mathcal{U})$ is monotonically nondecreasing and submodular. This useful submodular property allows us to approximately solve problem $(Q3)$. On the other hand, the relaxed capacity allocation subproblem $(Q2)$ becomes an LP, which can be efficiently solved; alternatively, we can also solve $(Q2)$ using a maximum flow algorithm (discussed at the end of Section VI-A).

B. Network Flow Formulation

In this subsection, we introduce a novel network flow reformulation of problem $(Q2)$. The purpose of this reformulation is two-fold: i) we will use it to prove that the objective function of the relaxed placement subproblem $(Q3)$ is submodular; ii) we will leverage it to develop a combinatorial algorithm for problem $(Q2)$, which is also part of an approximation algorithm we will propose for the original VPCA problem.

For problem $(Q2)$, we reformulate a network flow problem by constructing a directed graph $Z = (\mathcal{N}, \mathcal{L})$ as follows. The set of vertices $\mathcal{N}$ consists of the following: an artificial source vertex s, set $\mathcal{N}_{\mathcal{F}}$ consisting of flow-vertices f each corresponding to flow $f \in \mathcal{F}$, set $\mathcal{N}_{\mathcal{V}}$ consisting of node-vertices v each corresponding to node $v \in \mathcal{V}$, and set $\mathcal{N}_{\mathcal{V}'}$ consisting of node-vertices v' each corresponding to node $v \in \mathcal{V}$. Hence, $\mathcal{N} = \{s\} \cup \mathcal{N}_{\mathcal{F}} \cup \mathcal{N}_{\mathcal{V}} \cup \mathcal{N}_{\mathcal{V}'}$, where $\mathcal{N}_{\mathcal{V}}$ consists of the sinks. Let (x, y) be an edge in $\mathcal{L}$, which is from $x \in \mathcal{N}$ to $y \in \mathcal{N}$. The set of edges $\mathcal{L}$ consists of the following: set $\mathcal{L}_1$ consisting of edges (s, f) connecting the source vertex s to each flow-vertex $f \in \mathcal{N}_{\mathcal{F}}$, set $\mathcal{L}_2$ consisting of edges (f, v') connecting each flow-vertex $f \in \mathcal{N}_{\mathcal{F}}$ to each node-vertex $v' \in \mathcal{N}_{\mathcal{V}'}$ corresponding to a node $v \in \mathcal{V}_f$, set $\mathcal{L}_3$ consisting of edges (v', v) connecting each node-vertex $v' \in \mathcal{N}_{\mathcal{V}'}$ to its corresponding node-vertex $v \in \mathcal{N}_{\mathcal{V}}$. We use $c(x, y)$ to denote the capacity of edge (x, y). Hence, $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$. An edge $(s, f) \in \mathcal{L}_1$ has capacity λ_f; an edge $(f, v') \in \mathcal{L}_2$ has capacity λ_f; an edge $(v', v) \in \mathcal{L}_3$ has capacity c_v. Fig. 2 presents an example of the constructed graph Z for the network in Fig. 1.

Next, we describe flows over graph Z. Consider functions $\varphi(x, y) : \mathcal{N} \times \mathcal{N} \rightarrow \mathbb{R}_+$, where $\mathbb{R}_+$ is the set of non-negative real numbers. We define $\Phi(\mathcal{X}, \mathcal{Y}) \triangleq \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \varphi(x, y)$ for $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}$. An s-$\mathcal{V}$ flow is a function $\varphi(x, y) : \mathcal{N} \times \mathcal{N} \rightarrow \mathbb{R}_+$ such that the following is satisfied:

1) Capacity constraints: $\varphi(x, y) \leq c(x, y)$ for all pairs $(x, y) \in \mathcal{N} \times \mathcal{N}$. (Note that $c(x, y) = 0$ if $(x, y) \notin \mathcal{L}$.)
2) Flow conservation: the net-flow at every non-source non-sink vertex $x \in \mathcal{N} \backslash (\{s\} \cup \mathcal{N}_{\mathcal{V}})$ is zero, i.e., $\Phi(\mathcal{N}, \{x\}) - \Phi(\{x\}, \mathcal{N}) = 0$.
3) Positive incoming flow: the net-flow at the source s is non-positive, i.e., $\Phi(\mathcal{N}, \{s\}) - \Phi(\{s\}, \mathcal{N}) \leq 0$.

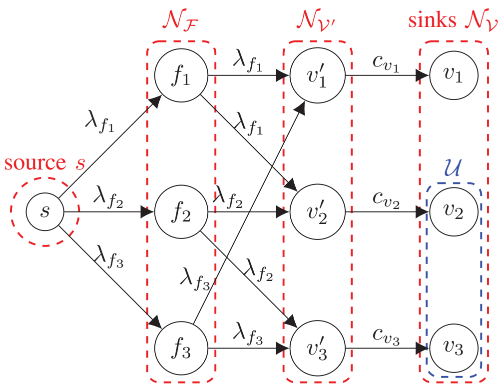


Fig. 2: An example of the constructed graph Z for the network in Fig. 1, where $\mathcal{F} = \{f_1, f_2, f_3\}$, $\mathcal{V} = \{v_1, v_2, v_3\}$, $\mathcal{V}_{f_1} = \{v_1, v_2\}$, $\mathcal{V}_{f_2} = \{v_2, v_3\}$, and $\mathcal{V}_{f_3} = \{v_1, v_3\}$

4) Positive outgoing flow: the net-flow at every sink $t \in \mathcal{N}_\mathcal{V}$ is non-negative, i.e., $\Phi(\mathcal{N}, \{t\}) - \Phi(\{t\}, \mathcal{N}) \geq 0$.

Let $\overline{\mathcal{F}}$ be the set of all s-$\mathcal{V}$ flows over Z.

For a subset of sinks[2] $\mathcal{U} \subseteq \mathcal{N}_\mathcal{V}$, we define

$$F(\mathcal{U}) \triangleq \max_{\varphi \in \overline{\mathcal{F}}} (\Phi(\mathcal{N}, \mathcal{U}) - \Phi(\mathcal{U}, \mathcal{N})), \tag{8}$$

which is the maximum total net-flow at the sinks in $\mathcal{U}$. The maximum net-flow problem is to find an s-$\mathcal{V}$ flow (i.e., function φ) that achieves the maximum in (8). In Lemma 1, we show the equivalence between the capacity allocation subproblem ($Q2$) and the maximum net-flow problem (8).

Lemma 1. *The capacity allocation subproblem* ($Q2$) *is equivalent to the maximum net-flow problem* (8)*. Hence, for any given $\mathcal{U} \subseteq \mathcal{V}$, the optimal value of problem* ($Q2$) *is equal to the maximum total net-flow at the sinks in $\mathcal{U} \subseteq \mathcal{N}_\mathcal{V}$ of the associated graph Z, i.e.,*

$$R_3(\mathcal{U}) = F(\mathcal{U}). \tag{9}$$

VI. Proposed Algorithms

In this section, we design two efficient algorithms that can achieve constant approximation ratios for the VPCA problem ($P1$). The main idea is to utilize the relaxation introduced in the previous section, which allows partially processed flows to be counted. By doing so, we can show that the relaxed placement subproblem is submodular based on the network flow reformulation of the relaxed capacity allocation subproblem. In this case, the relaxed placement subproblem can be approximately solved using efficient greedy algorithms. Moreover, the relaxed allocation subproblem becomes a Linear Program (LP), which can also be solved efficiently in polynomial time. However, the solution to the relaxed problem is for the case where any fraction of the processed flows is counted. In order to obtain a solution for the original VPCA problem ($P1$), where only the fully processed flows are counted, we propose two approximation algorithms by modifying the solution to the relaxed capacity allocation subproblem: the first one is based on a maximum flow algorithm, and the second one is based on a greedy algorithm.

[2]Note that each node $v \in \mathcal{V}$ corresponds to a sink in $\mathcal{N}_\mathcal{V}$. Hence, by slightly abusing the notations, for any $\mathcal{U} \subseteq \mathcal{V}$, we also use $\mathcal{U}$ to denote the corresponding subset of sinks in $\mathcal{N}_\mathcal{V}$.

Algorithm 1 The RP-MCA and RP-GCA algorithms

Input: set of nodes $\mathcal{V}$, set of flows $\mathcal{F}$, node capacities, node costs, flow rates, and budget B.
Output: set of VNF-nodes $\mathcal{U}$, capacity allocation $\boldsymbol{\lambda}$.
1: **Relaxed Problem**: relax function $J_1(\mathcal{U}, \boldsymbol{\lambda})$ to become $R_1(\mathcal{U}, \boldsymbol{\lambda})$;
2: **Placement Subproblem**: solve problem ($Q3$) using the SG algorithm or the EG algorithm, described in Section VI-A, to obtain $\mathcal{U}$.
3: **Capacity Allocation**: use either the MCA algorithm (Algorithm 2) or the GCA algorithm (Algorithm 3) to obtain capacity allocation $\boldsymbol{\lambda}$.

We use RP-MCA and RP-GCA to denote the algorithms we develop by combining the Relaxed Placement with the Maximum-flow-based Capacity Allocation and the Greedy Capacity Allocation, respectively. We show that the RP-MCA and RP-GCA algorithms achieve an approximation ratio of $\frac{1}{2}(1 - 1/e)$ and $\frac{1}{3}(1 - 1/e)$, respectively. We describe the algorithms in a unified framework presented in Algorithm 1. The difference is in the capacity allocation subproblem (line 3), where RP-MCA algorithm uses a Max-flow-based Capacity Allocation (MCA) algorithm presented in Algorithm 2, while RP-GCA algorithm uses a Greedy Capacity Allocation (GCA) algorithm presented in Algorithm 3.

A. *Proposed Placement Algorithms*

In this subsection, we first prove in Lemma 2 that the objective function $R_3(\mathcal{U})$ of the relaxed placement subproblem ($Q3$) is monotonically nondecreasing and submodular. Then, using the property of submodularity, we propose two greedy algorithms for solving the placement subproblem.

Lemma 2. *The function $R_3(\mathcal{U})$ is monotonically nondecreasing and submodular.*

Because of this useful submodular property, problem ($Q3$) can be approximately solved using efficient greedy algorithms. Next, we consider two cases of problem ($Q3$): uniform VNF-node costs (Case I, a special case) and heterogeneous VNF-node costs (Case II, a general case).

In Case I, the VNF-nodes have uniform costs, i.e., $b_v = b$ for all $v \in \mathcal{V}$. Then, the budget constraint (3) can be expressed as a cardinality constraint, i.e., $|\mathcal{U}| \leq k$, where $k = \lfloor B/b \rfloor$. In this case, we can use a simple *Submodular Greedy* (SG) algorithm to approximately solve problem ($Q3$). In the SG algorithm, we start with an empty solution of VNF-nodes $\mathcal{U}$; in each iteration, we add a node that has the maximum marginal contribution to $\mathcal{U}$, i.e., a node that leads to the largest increase in the value of the objective function. We repeat the above procedure until k VNF-nodes have been selected. This solution has been shown to achieve an approximation ratio of $(1-1/e)$ [16]. However, this algorithm does not guarantee to have the same approximation ratio for the case of heterogeneous VNF-node costs [17].

In Case II, the VNF-nodes have heterogeneous costs, i.e., the costs of VNF-nodes are different. For this case, an *Enumeration-based Greedy* (EG) algorithm has been proposed in [17], which can be shown to achieve the same approximation ratio of $(1-1/e)$, but with a higher running time complexity compared to the SG algorithm. The EG algorithm has two phases. In Phase I, it samples all node subsets of cardinality one or two that satisfy the budget constraint, picks the one with the largest value of the objective function R_3, and stores this temporary solution in $\mathcal{U}_1$. In Phase II, the algorithm samples all node subsets of cardinality three and augments each of these subsets with nodes that maximize the relative marginal contribution $(R_3(\mathcal{V}' \cup \{u\}) - R_3(\mathcal{V}'))/b_u$, in a greedy manner. The budget constraint must also be satisfied throughout this procedure. Then, it selects the augmented subset with the largest value of the objective function R_3 and stores it in $\mathcal{U}_2$. The final solution will be the better one between $\mathcal{U}_1$ and $\mathcal{U}_2$, i.e., the one that achieves a larger value of the objective function R_3.

Note that although the value of function $R_3(\mathcal{U})$ can be obtained using an LP solver, we can alternatively compute it using the network flow formulation presented in Section V-B as follows. For the constructed graph Z, we connect all the sink vertices corresponding to nodes $\mathcal{U}$ to an artificial sink vertex d. Then, the value of $R_3(\mathcal{U})$ is the maximum flow from vertex s to vertex d in graph Z, which can be computed using several efficient algorithms (see, e.g., [19]). In Lemma 3, we restate the results of [16], [17] about the approximation ratio of the SG and EG algorithms.

Lemma 3. *Both the SG and EG algorithms achieve an approximation ratio of* $(1-1/e)$.

Proof. The proofs can be found in [16] and [17] for the SG algorithm and the EG algorithm, respectively. ☐

B. Proposed Capacity Allocation Algorithms

While the solution of problem $(Q3)$ allows partially processed flows to be counted, only fully processed flows will be counted in the original problem $(P1)$. To that end, we propose two algorithms to modify the capacity allocation of VNF-nodes $\mathcal{U}$ so as to ensure fully processed flows and provide certain performance guarantees. The first algorithm is based on the network flow formulation, and the second one is based on a simple greedy approach. We develop these algorithms by modifying two algorithms for the multiple knapsack problem with assignment restrictions (MKAR) [20]. However, we want to point out that there is a key difference between our studied VPCA problem and the MKAR problem: in the VPCA problem, a flow can be split and assigned to more than one VNF-node, while in the MKAR problem, an item (corresponding to a flow in our problem) cannot be split and must be assigned to at most one knapsack (corresponding to VNF-node in our problem). Because of this key difference, an optimal solution for the VPCA problem generally has a larger value compared to that of the MKAR problem. Therefore, the algorithms developed for the MKAR problem need to be modified so as to yield a better performance.

Algorithm 2 The MCA algorithm

Input: set of VNF-nodes $\mathcal{U}$, set of flows $\mathcal{F}_\mathcal{U}$, flow rates, and VNF-node capacities.
Output: Capacity allocation $\boldsymbol{\lambda}$.
Phase I:
Obtain a basic optimal solution $\boldsymbol{\lambda}_\mathcal{U}$;
$y_f^v \triangleq \lambda_f^v / \lambda_f$, for all λ_f^v in $\boldsymbol{\lambda}_\mathcal{U}$;
Assign each flow f with $y_f^v = 1$ to VNF-node v;
Construct G' for the unassigned flows with positive y_f^v;
while G' is not empty **do**
  while there is a singlton VNF-node in G' **do**
    Perform the rounding in Step 1;
  end while
  Perform the rounding in Step 2;
end while
Phase II:
for each flow f in $\mathcal{F}_\mathcal{U}$ that is not assigned yet **do**
  if $c'(\mathcal{U}_f) \geq \lambda_f$ **then**
    Assign flow f to a subset of VNF-nodes in $\mathcal{U}_f$;
  end if
end for

First, we introduce some additional notations for the algorithms that will be described soon. We use $\mathcal{U}_f$ to denote the nodes on the path of flow f that are included in $\mathcal{U}$, i.e., $\mathcal{U}_f = \mathcal{V}_f \cap \mathcal{U}$. Let c'_v denote the remaining capacity of VNF-node v, and let $c'_{\mathcal{U}_i}$ denote the total remaining capacity of the set of VNF-nodes in $\mathcal{U}_i$, i.e., $c'_{\mathcal{U}_i} = \sum_{v \in \mathcal{U}_i} c'_v$. In what follows, we will introduce the MCA algorithm and the GCA algorithm.

1) Maximum-flow-based Capacity Allocation (MCA): We first present the MCA algorithm (Algorithm 2), a capacity allocation algorithm based on the network flow formulation. The MCA algorithm has two phases. In Phase I, MCA makes allocation decisions by rounding a fractional flow assignment obtained by solving problem $(Q2)$; in Phase II, the remaining VNF-node capacities are allocated in a greedy manner.

Phase I: Let $\boldsymbol{\lambda}_\mathcal{U}$ be a flow assignment obtained from an optimal basic solution[3] of problem $(Q2)$, which can be obtained by solving a maximum flow problem as discussed earlier. We use $y_f^v \triangleq \lambda_f^v / \lambda_f$ to denote the fraction of flow f assigned to VNF-node v in the obtained solution $\boldsymbol{\lambda}_\mathcal{U}$. The algorithm begins with a temporary assignment of every flow f with $y_f^v = 1$ to the corresponding VNF-node v. For the remaining flows, we do the following. Let $G' = (\mathcal{F}', \mathcal{V}', \mathcal{E}')$ be a bipartite graph constructed as follows. For each $\lambda_f^v \in \boldsymbol{\lambda}_\mathcal{U}$, if $0 < y_f^v < 1$, we add a flow vertex f to the set $\mathcal{F}'$, a VNF-node vertex v to the set $\mathcal{V}'$, and an edge, with weight y_f^v,

[3] A basic feasible solution is a solution that cannot be expressed as a convex combination of two feasible solutions.

connecting flow vertex f to VNF-node vertex v, to the set $\mathcal{E}'$. Note that graph G' cannot have a cycle because $\boldsymbol{\lambda}_{\mathcal{U}}$ is a basic feasible solution [20, Lemma 5]. After constructing graph G', we repeatedly apply the following two steps to graph G' until it becomes empty. As a result, the modified flow assignment y_f^v will become either zero or one.

Step 1: For each VNF-node $v \in \mathcal{V}'$ that has only one incident flow f (called a singleton VNF-node), we modify its capacity allocation as follows. Let r_v denote the total amount of flow rates assigned to VNF-node v and let r'_v be the portion of r_v contributed by fully assigned flows. Note that $r_v = r'_v + \lambda_f^v$. If $r'_v \geq \lambda_f^v$, then we set y_f^v to zero. Now, VNF-node v has no incident edges to it, so we remove it from G'. In this case, the value of solution $\boldsymbol{\lambda}_{\mathcal{U}}$ will be reduced by λ_f^v, which is no greater than $\frac{1}{2}r_v$. If $r'_v < \lambda_f^v$, then we unassign the flows temporarily assigned to VNF-node v and assign flow f to VNF-node v instead, i.e., set y_f^v to one, and cancel the other fractions of flow f assigned to other VNF-nodes. This is feasible because the rate of any flow is assumed to be no larger than the minimum VNF-node capacity. Then, we remove VNF-node v, flow f, and the associated edges from G'. In this case, the value of solution $\boldsymbol{\lambda}_{\mathcal{U}}$ will be reduced by at most r'_v, which is no greater than $\frac{1}{2}r_v$. We repeat Step 1 until no singleton VNF-node exists. Then, we go to Step 2.

Step 2: In this step, we will perturb the fractional values of some edges in G' to make one of them either zero or one. The perturbation is designed such that the capacity and assignment constraints are not violated and the total assigned traffic remains the same. We describe the perturbation procedure in the following. Consider a VNF-node $v_1 \in \mathcal{V}'$ that has a degree of at least two. Let (v_1, f_1) and (v_1, f_{k+1}) denote two of the incident edges to VNF-node v_1. Let $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ denote the longest paths starting from VNF-node v_1 through edges (v_1, f_1) and (v_1, f_{k+1}), respectively; such paths exist because G' is a forest. Here, we use y_i^j to denote the fractional value of flow i assigned to VNF-node j and use λ_j to denote the rate of flow j. Let $\boldsymbol{y_1} = (y_1^1, y_1^2, \dots, y_k^k)$ denote the fractional flow assignment on the edges of path $\boldsymbol{p_1}$, and let $f_1, \dots, f_k$ be the flow nodes of path $\boldsymbol{p_1}$. Similarly, let $\boldsymbol{y_2} = (y_{k+1}^1, y_{k+1}^{k+1}, \dots, y_{k+l}^{k+l-1})$ denote the fractional flow assignment on the edges of path $\boldsymbol{p_2}$, and let $f_{k+1}, \dots, f_{k+l}$ denote the flow nodes of path $\boldsymbol{p_2}$. We perturb $\boldsymbol{y_1}$ by adding to it $\boldsymbol{y'_1} = (\frac{\lambda_k}{\lambda_1}\epsilon, -\frac{\lambda_k}{\lambda_1}\epsilon, \frac{\lambda_k}{\lambda_2}\epsilon, \dots, -\frac{\lambda_k}{\lambda_{k-1}}\epsilon, \epsilon)$, and we perturb $\boldsymbol{y_2}$ by adding to it $\boldsymbol{y'_2} = (-\frac{\lambda_k}{\lambda_{k+1}}\epsilon, \frac{\lambda_k}{\lambda_{k+1}}\epsilon, -\frac{\lambda_k}{\lambda_{k+2}}\epsilon, \dots, \frac{\lambda_k}{\lambda_{k+l-1}}\epsilon, -\frac{\lambda_k}{\lambda_{k+l}}\epsilon)$. We increase ϵ until one fractional value y_f^v becomes zero or one, and if one, i.e., $y_f^v = 1$, then we assign flow f to the corresponding VNF-node v. An example to illustrate this step is shown in Fig. 3. In this new solution, at least one edge is removed from G'. We repeat the perturbation procedure until at least one VNF-node becomes a singleton, and then we go back to Step 1. If G' becomes empty, we start Phase II.

Phase II: We leverage the property that the traffic of a flow can be split and processed at multiple VNF-nodes. That is, after Phase I, we pick an unassigned flow f and check if the total remaining capacity of VNF-nodes $\mathcal{U}_f$ is no smaller than λ_f. If so, we split flow f so that the remaining capacities of some VNF-nodes in $\mathcal{U}_f$ can be used to fully process flow f and assign flow f to a subset of these VNF-nodes. We repeat this procedure until no more flow can be assigned.

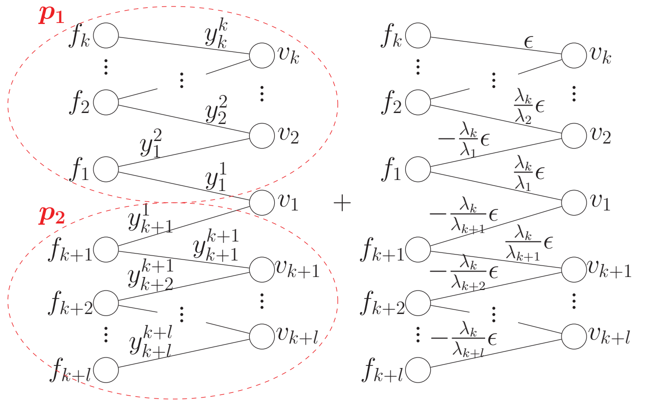


Fig. 3: An example of the edges perturbation

We use $OPT(Q2, \mathcal{U})$ to denote the total traffic assigned to a given set of VNF-nodes $\mathcal{U}$ by an optimal solution to problem (Q2). Also, we use $\pi_{\text{MCA}}^{\mathcal{U}}$ to denote the total traffic assigned to VNF-nodes $\mathcal{U}$ by the MCA algorithm. The approximation ratio of the MCA algorithm is stated in the following Lemma.

Lemma 4. *The MCA algorithm has an approximation ratio of* $1/2$, *i.e.,* $\pi_{MCA}^{\mathcal{U}} \geq \frac{1}{2}OPT(Q2, \mathcal{U})$.

Note that the $1/2$-approximation ratio of the MCA algorithm can be obtained by implementing Phase I only. Phase II improves the empirical performance as a result of splitting flows that are processed at multiple VNF-nodes, although the guaranteed approximation ratio remains the same.

2) Greedy Capacity Allocation (GCA): While the MCA algorithm achieves an approximation ratio of $1/2$, it has a relatively high complexity of $O(F^2V^2)$ (refer to Table I and the technical report [7] for the complexity analysis). This high complexity may render the MCA algorithm unsuitable for certain scenarios in practice. To that end, we propose the GCA algorithm, a simple greedy capacity allocation algorithm that has a much lower complexity of $O(FV)$. A lower complexity of the GCA algorithm is achieved at the cost of a slightly worse approximation ratio of $1/3$ (Lemma 5). However, the approximation ratio of the GCA algorithm can be improved to $2/5$ (Lemma 6) if an additional mild assumption (Assumption 1) holds. The GCA algorithm has two phases. In Phase I, we sort flows of $\mathcal{F}_{\mathcal{U}}$ in a nonincreasing order of their flow rates. Then, we iteratively go through the sorted list and assign each flow to any VNF-node in $\mathcal{U}_f$ if it has a sufficient capacity. In Phase II, the remaining capacities of the VNF-nodes can be allocated in a similar way to Phase II of the MCA algorithm by leveraging the property that a flow can be processed at multiple VNF-nodes. However, here the remaining flows need to be considered according to the order in the sorted list $\mathcal{F}_{\mathcal{U}}$. The GCA algorithm is presented in Algorithm 3.

Algorithm 3 The GCA algorithm

Input: set of VNF-nodes $\mathcal{U}$, set of flows $\mathcal{F}_\mathcal{U}$, flow rates, and VNF-node capacities.
Output: Capacity allocation $\boldsymbol{\lambda}$.
Sort flows $\mathcal{F}_\mathcal{U}$ in a noincreasing order of their flow rates;
Phase I:
for each flow f in the sorted set $\mathcal{F}_\mathcal{U}$ **do**
    if there is a VNF-node v in $\mathcal{U}_f$ such that $c'_v \geq \lambda_f$ **then**
        Set $\lambda_f^v = \lambda_f$;
        Set $c'_v = c'_v - \lambda_f$;
    end if
end for
Phase II:
for each flow f in the sorted set $\mathcal{F}_\mathcal{U}$ that is not assigned yet **do**
    if $c'(\mathcal{U}_f) \geq \lambda_f$ **then**
        Assign flow f to a subset of VNF-nodes in $\mathcal{U}_f$;
    end if
end for

In Lemma 5, we state the result about the approximation ratio of the GCA algorithm. We use $\pi^{\mathcal{U}}_{\text{GCA}}$ to denote the total traffic assigned to VNF-nodes $\mathcal{U}$ by the GCA algorithm.

Lemma 5. *The GCA algorithm has an approximation ratio of 1/3, i.e., $\pi^{\mathcal{U}}_{GCA} \geq \frac{1}{3}OPT(Q2,\mathcal{U})$.*

Further, we show in Lemma 6 that the approximation ratio of the GCA algorithm can be improved to $2/5$ when an additional mild assumption (Assumption 1) holds.

Assumption 1. *Assume that all the VNF-nodes in $\mathcal{U}$ have the same capacity and that every flow f in $\mathcal{F}_\mathcal{U}$ traverses at least two VNF-nodes in $\mathcal{U}$, i.e., $|\mathcal{V}_f \cap \mathcal{U}| \geq 2$.*

Lemma 6. *Suppose that Assumption 1 holds. Then, the GCA algorithm has an improved approximation ratio of $2/5$, i.e., $\pi^{\mathcal{U}}_{GCA} \geq \frac{2}{5}OPT(Q2,\mathcal{U})$.*

C. Main Results

We state our main results in Theorems 2 and 3.

Theorem 2. *The RP-MCA algorithm has an approximation ratio of $\frac{1}{2}(1-1/e)$ for problem $(P1)$.*

Proof. The proof combines the results in Lemmas 3 and 4. □

Theorem 3. *The RP-GCA algorithm has an approximation ratio of $\frac{1}{3}(1-1/e)$ for problem $(P1)$.*

Proof. The proof combines the results in Lemmas 3 and 5. □

Table I summarizes the complexity of all algorithms. In the literature, the complexity of algorithms for submodular functions is often measured using the number of function evaluations. The function evaluation itself is usually assumed to be conducted by an oracle, and thus its complexity is not taken into account [21]. We followed this approach here. Note that we can utilize other alternative algorithms to the EG algorithm to improve the running time substantially but with a slightly worse approximation ratio [17], [21]. We provide more discussions about the complexity analysis along with the tradeoff between the performance and complexity in the technical report [7].

Setting	Algorithm	Approximation	Complexity
Homogeneous VNF costs	RP-MCA	$\frac{1}{2}(1-1/e)$	$O(kV)^\dagger + O(F^2V^2)$
	RP-GCA	$\frac{1}{3}(1-1/e)$ $\frac{2}{5}(1-1/e)^*$	$O(kV)^\dagger + O(FV)$
Heterogeneous VNF costs	RP-MCA	$\frac{1}{2}(1-1/e)$	$O(V^5)^\dagger + O(F^2V^2)$
	RP-GCA	$\frac{1}{3}(1-1/e)$ $\frac{2}{5}(1-1/e)^*$	$O(V^5)^\dagger + O(FV)$

TABLE I: Approximation ratios and time complexities of the proposed algorithms. *These are the approximation results for the GCA algorithm when Assumption 1 holds. †This is the number of function evaluations used in the submodular optimization.

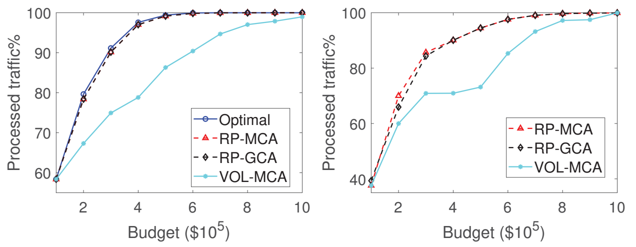
VII. Numerical Results

In order to evaluate the performance of the proposed algorithms, we first consider real-world network topologies and traffic statistics. We also extend the evaluations to synthesized networks consisting of a larger number of nodes and flows. We compare the proposed algorithms with the following algorithms: 1) optimal solution: we can solve problem $(P1)$ optimally using an ILP solver. However, this can be done for small instances only. 2) VOL-MCA [6]: this scheme selects the nodes with the highest traffic volume that traverses them. For the selected nodes, we allocate their capacity using the proposed MCA algorithm. We evaluate all algorithms based on the percentage of the processed traffic achieved by them, which is defined as the ratio between the total volume of the traffic processed by the VNF-nodes and the total traffic volume. The running time of the considered algorithms will also be presented. We run the simulations on a PC with i7 processor and 32GB physical memory.

A. Trace-driven Evaluation

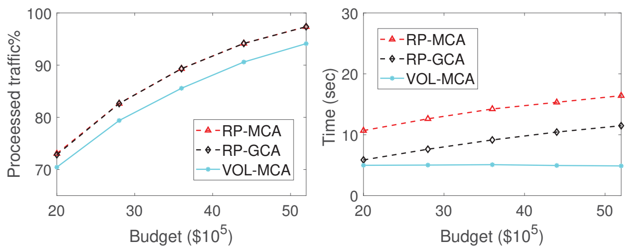
We consider the Abilene dataset [22] collected from an educational backbone network in North America. The network consists of 12 nodes and 144 flows. Each flow rate was recorded every five minutes for 6 months. Also, OSPF weights were recorded, which allows us to compute the shortest path of each flow based on these weights. In our experiments, we set the flow rate to the recorded value of the first day at 8:00 pm. The cost of a VNF-node is set to $\$100K$, and the processing capacity is set to 1 Gbps. We vary the total budget between $\$100K$ and $\$1M$. Since it takes too long for the ILP solver to find the optimal solution for 144 flows, we first consider a subset of 45 flows.

Fig. 4 shows the percentage of processed traffic for the considered algorithms. We can see that both the RP-MCA and RP-GCA algorithms perform almost the same as the optimal solution and have up to 20% improvement over the VOL-MCA algorithm. Note that as the budget increases, the total



(a) 45 flows (b) 144 flows

Fig. 4: Evaluation of Abilene dataset



(a) Processed traffic (b) Running time

Fig. 5: Evaluation on a large synthesized network consisting of 100 nodes

processed traffic increases under all the considered algorithms. However, while the proposed algorithms need a budget of $\$500K$ to process 45 flows, the VOL-MCA algorithm actually requires double of the budget (around $\$1M$). We make similar observations when we consider all the 144 flows in the Abilene dataset. In this case, it becomes infeasible to derive an optimal solution using ILP solvers.

B. Evaluation on Synthesized Network Settings

Next, we evaluate the algorithms for a larger synthesized network consisting of 100 nodes and 800 flows. We repeat each experiment for 10 times and present the average results. Fig. 5(a) shows that the proposed algorithms still exhibit a superior performance compared to the VOL-MCA algorithm. In Fig. 5(b), we observe that all the algorithms run very fast (i.e., finish in less than 20 seconds). The simulation results suggest that the proposed algorithms achieve a very good tradeoff between the performance and the running time.

VIII. Conclusion

In this paper, we studied the problem of deploying VNF-nodes and allocating their capacity. We showed how to overcome the non-submodularity of the problem by introducing a novel relaxation method. By utilizing a decomposition of the problem and a novel network flow reformulation, we were able to prove the submodularity of the relaxed placement subproblem and develop efficient algorithms with constant approximation ratios for the original problem. Through extensive evaluations using both traces and synthesized networks, we showed that the proposed algorithms have a performance close to the optimal solution and better than a state-of-the-art algorithm. In our future work, we will consider other important objectives such as delay and energy consumption. We shall also consider an online version of the problem where flows come and stay for a certain amount of time.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, vol. 48. sn, 2012.

[2] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing sdn upgrades in isp networks," in *Proceedings of IEEE INFOCOM*, 2017.

[3] Amdocs, "Bringing NFV to Life - Technological and Operational Challenges in Implementing NFV," *White paper*, 2016.

[4] 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 *Proceedings of IEEE INFOCOM*, 2017.

[5] T. Lukovszki, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *Journal of Parallel and Distributed Computing*, 2018.

[6] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of sdn in hybrid enterprise and isp networks," in *Proceedings of the Symposium on SDN Research*. ACM, 2016.

[7] G. Sallam and B. Ji, "Joint placement and allocation of virtual network functions with budget and capacity constraints," in *arXiv preprint https://arxiv.org/abs/1901.03931*, 2019.

[8] M. Shit, X. Lin, S. Fahmy, and D.-H. Shin, "Competitive online convex optimization with switching costs and ramp constraints," in *Proceedings of IEEE INFOCOM*, 2018.

[9] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," in *IEEE International Conference on Communications, ICC*, 2016.

[10] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the nfv service distribution problem," in *Proceedings of IEEE INFOCOM*, 2017.

[11] Y. Chen, J. Wu, and B. Ji, "Virtual network function deployment in tree-structured networks," in *IEEE 26th International Conference on Network Protocols (ICNP)*, 2018.

[12] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *Proceedings of IEEE INFOCOM*, 2018.

[13] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2015.

[14] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *Proceedings of IEEE INFOCOM*, 2018.

[15] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *IEEE ICDCS*, 2018.

[16] G. L. Nemhauser and L. A. Wolsey, "Maximizing submodular set functions: formulations and analysis of algorithms," in *North-Holland Mathematics Studies*. Elsevier, 1981, vol. 59, pp. 279–301.

[17] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information Processing Letters*, vol. 70, no. 1, pp. 39–45, 1999.

[18] M. Feldman and R. Izsak, "Constrained monotone function maximization and the supermodular degree," *arXiv preprint arXiv:1407.6328*, 2014.

[19] A. V. Goldberg and R. E. Tarjan, "Efficient maximum flow algorithms," *Communications of the ACM*, vol. 57, no. 8, pp. 82–89, 2014.

[20] M. Dawande, J. Kalagnanam, P. Keskinocak, F. S. Salman, and R. Ravi, "Approximation algorithms for the multiple knapsack problem with assignment restrictions," *Journal of combinatorial optimization*, vol. 4, no. 2, pp. 171–186, 2000.

[21] W. Li and N. Shroff, "Towards practical constrained monotone submodular maximization," *arXiv preprint arXiv:1804.08178*, 2018.

[22] "Abilene dataset," http://www.cs.utexas.edu/%7eyzhang/research/AbileneTM/, accessed: 2019-01-14.