Placement and Allocation of Virtual Network Functions: Multi-dimensional Case

Gamal Sallam, Zizhan Zheng, and Bo Ji

Abstract—**Network function virtualization (NFV) is an emerging design paradigm that replaces physical middlebox devices with software modules running on general purpose commodity servers. While gradually transitioning to NFV, Internet service providers face the problem of where to introduce NFV in order to make the most benefit of that; here, we measure the benefit by the amount of traffic that can be serviced through the NFV. This problem is non-trivial as it is composed of two challenging subproblems: 1) placement of nodes to support virtual network functions (referred to as VNF-nodes); and 2) allocation of the VNF-nodes resources to network flows; the two subproblems need to be considered jointly to satisfy the objective of serving the maximum amount of traffic. This problem has been studied recently but for the one-dimensional setting, where all network flows require one network function, which requires a unit of resource to process a unit of flow. In this work, we extend to the multi-dimensional setting, where flows can require multiple network functions, which can also require a different amount of each resource to process a unit of flow. The multi-dimensional setting introduces new challenges in addition to those of the one-dimensional setting (e.g., NP-hardness and non-submodularity) and also makes the resource allocation a multi-dimensional generalization of the generalized assignment problem with assignment restrictions. To address these difficulties, we propose a novel two-level relaxation method and utilize the primal-dual technique to design two approximation algorithms that achieve an approximation ratio of $\frac{(e-1)(Z-1)}{2e^2 Z(kR)^{1/(Z-1)}}$ and $\frac{(e-1)(Z-1)}{2e(Z-1+eZR^{1/(Z-1)})}$, where k (resp. R) is the number of VNF-nodes (resp. resources), and Z is a measure of the available resource compared to flow demand. Finally, we perform extensive trace-driven simulations to show the effectiveness of the proposed algorithms.**

I. Introduction

Network function virtualization (NFV) is a new design paradigm where network functions (e.g., firewall and load balancer) that traditionally run in dedicated hardware are now replaced by software modules hosted on general purpose commodity hardware [1]. Several advantages can be harnessed from this architecture such as reducing the deployment cost, increasing the agility, and improving the scalability. These advantages have encouraged several Internet service providers (ISPs) to consider this architecture, and some of them have already started the transition to NFV [2].

However, transitioning to NFV faces challenges from different perspectives. From network flows' perspective, each flow needs to be processed by certain types of network functions, and each network function requires a different amount of the resources at servers (e.g., CPU, memory). In addition, flows generally need all of their traffic to be fully processed by such functions to satisfy certain quality of services [3]. From ISPs' perspective, transitioning to NFV usually happens in multiple stages for several reasons such as the desire to utilize the already provisioned hardware, or budget limitations. Considering both of these two perspectives leads to an important question: under a limited budget, how to efficiently introduce NFV in each stage such that the total traffic of fully processed flows is maximized? To answer this question, we need to address two main issues: 1) where to place nodes that support NFV (called VNF-nodes) without exceeding the given budget? And 2) how to allocate the VNF-nodes resources to satisfy the requirements of network flows? We refer to this problem as joint VNF-nodes placement and resource allocation (VPRA).

Most of the previous work either does not have a limited budget (e.g., [4]) or relaxes the resources constraint (e.g., [3]). In [5], both the budget and resources constraints are considered, along with fully processed flows requirement. However, they consider a special case of the VPRA problem with the following characteristics: a) there is only one type of resource; b) all flows require the same network function; and c) the network function requires one unit of resource to process each unit of flows (we refer to this setting as basic-VPRA). Even under such a simplified setting, the basic-VPRA is already quite challenging. It is shown in [5] that the problem is not only NP-hard but also non-submodular (a property that generally leads to efficient solutions for similar problems (e.g., [3])). In this work, we take one step further and extend the basic-VPRA problem to consider the setting with multiple network functions, multiple resources, and heterogeneous resource requirements. We refer to this generalization as multi-dimensional VPRA (multi-VPRA).

We systematically study the challenges of the multi-VPRA problem and show that the difficulties introduced by the generalization call for different algorithms and design strategies. Specifically, we consider the placement subproblem and the allocation subproblem separately. We show that the placement subproblem even with removing the fully processed flow requirement is not easy to solve. Further, the resource allocation subproblem is a multi-dimensional generalization of the generalized assignment problem with assignment restrictions, which is a challenging problem [6]. To overcome the challenges of the placement subproblem, we introduce a two-level relaxation that allows us to draw a connection to the sequence submodular (also called string submodular)

This work was supported in part by the NSF under Grants CNS-1651947.
Gamal Sallam (tug43066@temple.edu) and Bo Ji (boji@temple.edu) are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, Zizhan Zheng (zzheng3@tulane.edu) is with the Department of Computer Science, Tulane University, New Orleans, LA.

978-1-7281-2700-2/19/$31.00 2019 © IEEE

theory [7], [8] and design an efficient placement algorithm. For the resource allocation subproblem, we utilize the primal-dual technique [9] to design two efficient resource allocation algorithms. We combine the placement algorithm with the resource allocation algorithms and develop approximation algorithms with performance guarantees for the original non-relaxed multi-VPRA problem.

Our main contributions are summarized as follows.

- First, we systematically study the challenges arising from the generalized multi-VPRA problem. In addition to the challenges faced by the basic-VPRA (such as NP-hardness and non-submodularity), we show that overcoming the non-submodularity of the placement subproblem is much harder and that the resource allocation subproblem is a multi-dimensional generalization of the generalized assignment problem with assignment restrictions, which is also more challenging.
- Second, we introduce a novel two-level relaxation method that enables us to convert the non-submodular placement subproblem into a sequence submodular optimization problem. Leveraging this useful property of sequence submodularity, we develop an efficient algorithm for the VNF-nodes placement. In addition, we utilize the primal-dual technique to design two efficient resource allocation algorithms.
- Third, we show that by combining the proposed placement algorithm and the two resource allocation algorithms, we can achieve an approximation ratio of $\frac{(e-1)(Z-1)}{2e^2 Z(kR)^{1/(Z-1)}}$ and $\frac{(e-1)(Z-1)}{2e(Z-1+eZR^{1/(Z-1)})}$ for the original non-relaxed multi-VPRA problem, respectively, where k (resp. R) is the number of VNF-nodes (resp. resources), and Z is a measure of the available resource compared to flow demand. When Z goes to infinity, the approximation ratio becomes $\frac{e-1}{2e^2}$ and $\frac{e-1}{2e^2+2e}$, respectively.
- Finally, we conduct extensive trace-driven simulations using Abilene dataset [10] as well as datasets from SNDlib [11] to evaluate the performance of the proposed algorithms.

II. Related Work

The placement problem has been considered in different domains such as NFV (e.g., [5]), SDN (e.g., [3]), and edge cloud computing (e.g., [12]). In NFV, several studies (e.g., [4], [13], [14]) consider the placement of a minimum number of VNF instances to cover all flows. A single type of network functions is considered in [4], [13], [15], [16], and the case of multiple network functions is considered in [14], [17], [18], [19], [20]. However, these work neglects either the budget constraint or the multi-dimensional resource allocation. The work in [21] considers the placement of middleboxes to make the shortest path between communicating pairs under a threshold. Again, this work does not consider multiple network functions or budget constraint. The closest work to ours is [5], where budget, resource, and fully processed flow constraints are considered, but it only considers one type of network functions and only a single type of resource.

In the SDN domain, the work in [3] considers the placement of SDN-enabled routers to maximize the total processed traffic. They consider a budget constraint but neglect the limited resources constraint. Similarly, in the work on edge cloud computing [14], although the budget and resource constraints are considered, their proposed solution is only for a special case, and the overall problem does not consider the multi-dimensional setting. To the best of our knowledge, the multi-dimensional setting has rarely been considered except in a limited number of studies. In [22], the authors consider multi-resource VNFs with a focus on the analysis of the vertical scaling (scaling up/down of some resources) and horizontal scaling (the number of VNFs instances). The work of [23] focuses only on request admission and routing. The work of [24] also considers the multi-resource setting, but the focus is on how to balance the load among the servers, taking into consideration the different demand of network functions for each resource. Our work considers the three constraints of budget, resource, and fully processed flows, as well as the multi-dimensional setting.

The concept of sequence (or string) submodularity is an extension of submodularity, which has been introduced recently in several studies (e.g., [7], [8], [25]). It models objective functions that depend on the sequence of actions. It has been utilized to design approximation algorithms for different applications such as online advertising [8]. To the best of our knowledge, we are the first to utilize the concept of sequence submodularity for the placement problem in NFV. Another concept is the primal-dual technique, which has been utilized extensively to design approximation algorithms for several problems [9]. We utilize this technique to design efficient algorithms for the multi-dimensional resource allocation subproblem.

III. System model

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. 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 at least one node in a subset of nodes $\mathcal{U} \subseteq \mathcal{V}$, i.e., $\mathcal{F}_{\mathcal{U}} = \{f \in \mathcal{F} \mid \mathcal{V}_f \cap \mathcal{U} \neq \emptyset\}$. When a node can 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.

We consider a set of network functions denoted by Φ. Each flow needs to be processed by one or more network functions. The set of network functions required by flow f is denoted by Φ_f. The set of flows that require network function $\phi \in \Phi$ is denoted as $\mathcal{F}(\phi)$. Each VNF-node $v \in \mathcal{V}$ can host one or more network functions. We use $\mathcal{R}$ to denote the set of resource types at VNF-nodes (e.g., memory and CPU), with $R = |\mathcal{R}|$. Each network function ϕ requires β_ϕ^r units of resource $r \in \mathcal{R}$ to process one unit of a network flow. 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 fully processed at VNF-nodes. Hence, when a flow traverses VNF-nodes and there are sufficient resources 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 fully processed traffic for a subset of VNF-nodes $\mathcal{U} \subseteq \mathcal{V}$ 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. However, there is a total amount of each resource available at the nodes, and the amounts could be different at different nodes. We use c_v^r to denote the total amount of resource r at node v. Then, the following constraints should be satisfied:

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

Also, we consider a limited budget B and assume that the cost for making node v a VNF-node is the same for all nodes, which is denoted by b. Let $k = \lfloor B/b \rfloor$. Then, the budget constraint can be expressed as a cardinality constraint, i.e.,

$$|\mathcal{U}| \leq k. \tag{3}$$

As a service provider with a limited budget, a plausible objective is to introduce NFV at nodes that would result in the maximum fully processed traffic. Therefore, we consider the problem of multi-dimensional VNF-nodes placement and resource allocation (multi-VPRA) with the objective of maximizing the total fully processed traffic ($J_1(\mathcal{U}, \boldsymbol{\lambda})$). The problem can be formulated as:

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

IV. Challenges of Multi-VPRA

In this section, we analyze the multi-VPRA problem and identify the main challenges posed by this problem. We first decompose the multi-VPRA problem into two subproblems: 1) placement, i.e., where to deploy VNF-nodes; 2) resource allocation of the VNF-nodes among flows. We will show the hardness of each subproblem and explain new challenges arising from the multi-dimensional generalization.

A. Decomposition

In this subsection, we present a decomposition of the multi-VPRA problem into placement and allocation 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 used to indicate that it is associated with a given set of VNF-nodes $\mathcal{U}$. Then, the resource 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 placement value function, which is the optimal value of problem ($P2$) for a given set of VNF-nodes $\mathcal{U}$. Then, the placement subproblem can be formulated as

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

Note that in order to solve subproblem ($P3$), we need to solve subproblem ($P2$) first to find the optimal $\boldsymbol{\lambda}$ for a given set of VNF-nodes $\mathcal{U}$.

B. Hardness

In [5, Theorem 1], it is shown that for the basic-VPRA problem, both subproblems ($P2$) and ($P3$) are NP-hard. The NP-hardness results can be easily extended to the multi-dimensional case considered here. Therefore, we simply state the hardness results in the following lemma without proofs.

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

In addition, the placement subproblem of the basic-VPRA has been shown to be non-submodular [5, Section IV. B]. Similarly, the non-submodularity result can also be easily extended to the multi-dimensional case. In order to develop efficient algorithms for the basic-VPRA, the work of [5] employs a relaxation of the problem that allows partially processed flows to be counted in the objective function. The relaxation allows one to prove submodularity of the placement subproblem and to design efficient algorithms for the basic-VPRA. However, in the sequel, we will explain that the same framework and algorithms cannot be applied directly to solve the multi-VPRA problem.

The first challenge is that a similar relaxation of the basic-VPRA does not admit an efficient placement algorithm with performance guarantees for the multi-VPRA problem. The reason is that the objective function of the relaxed placement subproblem of the basic-VPRA problem can be shown to be equivalent to the maximum flow problem, which can be proved to be submodular. In contrast, the objective function of the relaxed placement subproblem of the multi-VPRA problem, to the best of our knowledge, can only be evaluated using Linear Programming, which does not provide us with enough insights that can be utilized to prove or disprove submodularity. The second challenge is that the resource allocation algorithms proposed for the basic-VPRA consider only a single resource and cannot be utilized to provide performance guarantees for the multi-VPRA problem, where multiple resources have to be considered during the resource allocation.

In order to address these new challenges, we introduce a novel two-level relaxation method: (i) we allow partially processed flows as in [5], and (ii) we consider an approximate version of the resource allocation subproblem. This new relaxation method enables us to make a connection between the relaxed placement subproblem and sequence submodular theory and design efficient placement algorithms. For the resource allocation, we design two resource allocation algorithms both based on the primal-dual technique. Not only the proposed placement and resource allocation algorithms can properly handle the multi-dimensional setting, but they also guarantee a constant approximation ratio for the original non-relaxed multi-VPRA problem.

V. Relaxed Multi-VPRA

In this section, we present the two-level relaxation of the multi-VPRA problem. In the first-level, we allow partially processed flows to be counted in the objective function, and in this case we use $R_1(\mathcal{U}, \boldsymbol{\lambda})$ to denote the relaxed objective function (defined in Eq. (4)). In the second-level, instead of evaluating function $R_1(\mathcal{U}, \boldsymbol{\lambda})$ for a set of nodes $\mathcal{U}$ together, we allow the algorithm to consider a specific ordering of nodes and evaluate the objective function on a node-by-node basis. Apparently, the first-level relaxation does not decrease the total traffic that can be assigned to a given set of VNF-nodes $\mathcal{U}$. In contrast, the second-level relaxation results in an approximate version of the resource allocation subproblem, and thus, there is a loss in the amount of processed traffic. However, we will prove that the loss is at most $1/2$ of the optimal. In addition, through simulation results, we will show that the loss due to the second-level relaxation is negligible. The purpose of this two-level relaxation is to draw a connection to the sequence submodular theory, which enables us to design efficient algorithms with provable performance guarantees.

A. First-level Relaxation

We first introduce the first-level relaxation, which allows partially processed flows to be counted. In this case, 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:

$$R_1(\mathcal{U}, \boldsymbol{\lambda}) \triangleq \sum_{f \in \mathcal{F}} \sum_{v \in \mathcal{V}_f \cap \mathcal{U}} \lambda_f^v. \tag{4}$$

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

$$\sum_{v \in \mathcal{U}} \lambda_f^v \le \lambda_f, \quad \forall f \in \mathcal{F}. \tag{5}$$

Then, after the first-level relaxation, problem $(P1)$ becomes

$$\begin{aligned} \underset{\mathcal{U} \subseteq \mathcal{V}, \boldsymbol{\lambda}}{\text{maximize}} \quad & R_1(\mathcal{U}, \boldsymbol{\lambda}) \\ \text{subject to} \quad & (2), (3), \text{ and } (5). \end{aligned} \tag{$Q1$}$$

Next, we explain why we need the second-level relaxation for solving the multi-VPRA problem efficiently. Similar to the decomposition of problem $(P1)$, we also decompose problem $(Q1)$ 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 $\boldsymbol{\lambda}$ that satisfy the resources constraint (2) and the flow rate constraint (5), 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 to indicate that $\mathcal{U}$ is not a decision variable. Then, the resource 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 $\mathcal{U}$. The function $R_3(\mathcal{U})$ is also called the placement value function, and the placement subproblem can be formulated as

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

Unlike the relaxed placement subproblem of the basic-VPRA problem, which has been proven to be submodular, the submodularity of the relaxed placement subproblem $(Q3)$ of the multi-VPRA remains unknown as explained earlier. Driven by this observation, in the next subsection we introduce another level of relaxation, which enables us to draw a connection to the sequence submodular theory.

B. Second-level Relaxation

In the second-level relaxation, instead of solving subproblem $(Q2)$ to obtain the optimal solution $R_3(\mathcal{U})$ for a set of nodes $\mathcal{U}$, we consider a specific ordering of nodes $\mathcal{U}$ and solve for each node one-by-one according to their order (which will be explained soon in Algorithm 1). First, we provide some notations. Let $(v_1, v_2, \ldots, v_k)$ be a sequence of nodes selected over k steps, where $v_i \in \mathcal{V}$ is selected in the i-th step. Let the set of all possible sequences of nodes be $\mathcal{V}^* = \{(v_1, v_2, \ldots, v_k) \mid k = 0, 1, \ldots, |\mathcal{V}| \text{ and } v_i \in \mathcal{V}\}$. For two sequences $A_1 = (v_1^{a_1}, v_2^{a_1}, \ldots, v_{k_1}^{a_1})$ and $A_2 = (v_1^{a_2}, v_2^{a_2}, \ldots, v_{k_2}^{a_2})$ in $\mathcal{V}^*$, we define a concatenation of A_1 and A_2 as:

$$A_1 \oplus A_2 = (v_1^{a_1}, v_2^{a_1}, \ldots, v_{k_1}^{a_1}, v_1^{a_2}, v_2^{a_2}, \ldots, v_{k_2}^{a_2}).$$

Also, we say that $A_1 \preceq A_2$ if we can write A_2 as $A_1 \oplus O$ for some $O \in \mathcal{V}^*$. By slightly abusing the notation, we use $|A|$ to denote the number of elements in sequence A.

We use $\bar{\lambda}_v$ to denote the total flow assigned to node v and use $\bar{\boldsymbol{\lambda}} = \{\bar{\lambda}_v, \ \forall \ v \in \mathcal{V}\}$ to denote a given feasible resource allocation for nodes $\mathcal{V}$. Consider any node v', we define an optimal fractional resource allocation of node v' given a fixed

Algorithm 1 Iterative resource allocation

Input: sequence of nodes S, set of flows $\mathcal{F}$, amount of resources c_v^r, flow rates λ_f, and flow demands β_f^r
Output: resource allocation $\bar{\boldsymbol{\lambda}}$

Initialize: set each $\bar{\lambda}_v \in \bar{\boldsymbol{\lambda}}$ to zero
for $i = 1$ to $|S|$ **do**
    Solve problem (6) for node v_i given $\bar{\boldsymbol{\lambda}}$
    Update the value of $\bar{\lambda}_{v_i} \in \bar{\boldsymbol{\lambda}}$ according to the solution of problem (6)
end for

resource allocation $\bar{\boldsymbol{\lambda}}$ of all other nodes to be the solution of the following problem:

$$
\begin{aligned}
&\underset{\boldsymbol{\lambda}}{\text{maximize}} \quad \sum_{f \in \mathcal{F}} \lambda_f^{v'} \\
&\text{subject to} \\
&\sum_{v \in \mathcal{V} \cap \mathcal{V}_f} \lambda_f^v \leq \lambda_f, && \forall f \in \mathcal{F}, \\
&\lambda_f^v = 0, && \forall f \in \mathcal{F} \text{ and } v \notin \mathcal{V} \cap \mathcal{V}_f, \\
&\sum_{\phi \in \Phi} \beta_\phi^r \sum_{f \in \mathcal{F}(\phi)} \lambda_f^v \leq c_v^r, && \forall r \in \mathcal{R} \text{ and } v \in \mathcal{V}, \\
&\sum_{f \in \mathcal{F}_v} \lambda_f^v = \bar{\lambda}_v, && \forall v \in \mathcal{V} \setminus \{v'\},
\end{aligned}
\tag{6}
$$

where the last constraint is to maintain the given resource allocation $\bar{\boldsymbol{\lambda}}$ of nodes $\mathcal{V} \setminus \{v'\}$.

Now, consider sequence $S \in \mathcal{V}^*$, the resource allocation of nodes S is implemented as described in Algorithm 1. Basically, Algorithm 1 starts by initializing the total traffic assigned to each node to zero (i.e., set each $\bar{\lambda}_v \in \bar{\boldsymbol{\lambda}}$ to zero), and then iterates over nodes in sequence S according to their order. In iteration i, it computes the optimal resource allocation of node v_i by solving problem (6) given $\bar{\boldsymbol{\lambda}}$, and then update $\bar{\lambda}_{v_i} \in \bar{\boldsymbol{\lambda}}$ according to the obtained solution. We define function $R_4(S)$ to be the total traffic assigned by Algorithm 1 for nodes in sequence S. Then, the relaxed version of problem $(Q3)$ becomes:

$$
\begin{aligned}
&\underset{S \in \mathcal{V}^*}{\text{maximize}} \quad R_4(S) \\
&\text{subject to } |S| \leq k.
\end{aligned}
\tag{$Q4$}
$$

Next, we will show that function $R_4(S)$ is a $1/2$ approximation of function $R_3(\mathcal{U})$ as long as sequence S is one of the permutations of set $\mathcal{U}$. This ensures that an optimal solution for problem $(Q4)$ is $1/2$ approximation of the solution of problem $(Q3)$. Moreover, in the next section, we will utilize the relaxed problem $(Q4)$ to design efficient algorithms for the multi-VPRA problem $(P1)$.

In the following lemma, we present the approximation ratio of Algorithm 1.

Lemma 2. *For a given set of nodes $\mathcal{U}$, let $\mathcal{P}(\mathcal{U})$ be the set of all permutations of nodes $\mathcal{U}$. Then, for any $S \in \mathcal{P}(\mathcal{U})$, we have that $R_4(S) \geq \frac{1}{2} R_3(\mathcal{U})$.*

Proof. Let $\mathcal{F}' \subseteq \mathcal{F}$ denote the set of (partially or fully) unsatisfied flows at the end of Algorithm 1, and their traffic rate to be only the remaining traffic rate at the end of Algorithm 1. We use OPT($\mathcal{U}$) to denote the optimal resource allocation of VNF-nodes $\mathcal{U}$ and OPT($\mathcal{U}|\mathcal{F}'$) to denote the optimal resource allocation of VNF-nodes $\mathcal{U}$ given only the subset of flows $\mathcal{F}'$. We also use $\bar{\lambda}_{v_i}^S$ to denote the total traffic assigned to node v_i by Algorithm 1 when evaluating $R_4(S)$

The maximum traffic that can be assigned by any algorithm to VNF-nodes $\mathcal{U}$ has the following upper bound:

$$
\begin{aligned}
\text{OPT}(\mathcal{U}) &\overset{\text{(a)}}{\leq} R_4(S) + \text{OPT}(\mathcal{U}|\mathcal{F}') \\
&\leq R_4(S) + \sum_{i=1}^{|S|} \text{OPT}(\{v_i\}|\mathcal{F}') \\
&\overset{\text{(b)}}{\leq} R_4(S) + \sum_{i=1}^{|S|} \bar{\lambda}_{v_i}^S \\
&= R_4(S) + R_4(S) \\
&= 2R_4(S).
\end{aligned}
\tag{7}
$$

The upper bound in (a) basically combines what can be assigned from only a subset of flows $\mathcal{F}'$ to nodes $\mathcal{U}$, which is OPT($\mathcal{U}|\mathcal{F}'$), plus everything else, which is $R_4(S)$. Adding together OPT($\mathcal{U}|\mathcal{F}'$) and $R_4(S)$ should give an upper bound to OPT($\mathcal{U}$). Next, we show that (b) holds. Consider sequence S, when solving problem (6) for node v_i, we only need to worry about satisfying the equality constraints of nodes $(v_1, \ldots, v_{i-1})$, which are the nodes that have been assigned positive traffic from the previous steps of the algorithm. All other nodes have been only assigned zero traffic from the initialization step and can be satisfied without affecting the value of the objective function. However, flows $\mathcal{F}'$ is the set of unsatisfied flows at the end of the algorithm and after assigning $\bar{\lambda}_{v_i}^S$ to each node v_i in S, and so satisfying the equality constraint of all nodes in S, not just $(v_1, \ldots, v_{i-1})$. Therefore, OPT($\{v_i\}|\mathcal{F}'$) should be a feasible solution to node v_i when solving $R_4(S)$, so (b) holds. □

Based on Algorithm 1, for any two sequences A_1 and A_2 in $\mathcal{V}^*$ such that $A_1 \preceq A_2$, nodes in A_1 will be considered first when solving $R_4(A_1)$ and $R_4(A_2)$, and in the same order. So, the amount of assigned traffic to nodes A_1 will be the same for the two sequences A_1 and A_2. We state that in the following corollary.

Corollary 1. *For any two sequences A_1 and A_2 in $\mathcal{V}^*$ such that $A_1 \preceq A_2$, the amount of traffic assigned to each node in A_1 when solving $R_4(A_1)$ and $R_4(A_2)$ is the same.*

VI. Proposed Algorithms

In this section, we design two algorithms that approximately solve the multi-VPRA problem $(P1)$. The main idea is to apply the two-level relaxation introduced in the previous section on the original non-relaxed problem $(P1)$. By doing so, we can show that the objective function of the relaxed placement subproblem $(Q4)$ is sequence submodular (to be defined later).

Algorithm 2 The SSG-PRA and SSG-NRA algorithms

Input: set of nodes $\mathcal{V}$, set of flows $\mathcal{F}$, amount of resources, flow rates, flows demand, and budget B

Output: set of VNF-nodes $\mathcal{U}$ and resource allocation $\boldsymbol{\lambda}$

1: **Relaxed Problem**: (first-level relaxation) relax function $J_1(\mathcal{U}, \boldsymbol{\lambda})$ to become $R_1(\mathcal{U}, \boldsymbol{\lambda})$, and (second-level relaxation) relax function $R_3(\mathcal{U})$ to function $R_4(S)$

2: **Placement Subproblem**: solve problem $(Q4)$ using the greedy algorithm (Algorithm 3) to obtain S

3: **Resource Allocation**: use either the PRA algorithm (Algorithm 4) or the NRA algorithm (Algorithm 5) to obtain resource allocation $\boldsymbol{\lambda}$ for nodes S

In this case, the relaxed placement subproblem can be approximately solved using an efficient greedy algorithm. 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 multi-VPRA problem $(P1)$, where only the fully processed flows are counted, we propose two approximation algorithms based on the primal-dual technique.

We use SSG-PRA and SSG-NRA to denote the algorithms we develop by combining the Sequence Submodular Greedy placement with the Primal-dual-based Resource Allocation and the Node-based Resource Allocation, respectively. We describe the algorithms in a unified framework presented in Algorithm 2. The difference is in the resource allocation subproblem (line 3), where SSG-PRA algorithm uses a Primal-dual-based Resource Allocation (PRA) algorithm presented in Algorithm 4, while SSG-NRA algorithm uses a Node-based Resource Allocation (NRA) algorithm presented in Algorithm 5. We show that the SSG-PRA and SSG-NRA algorithms achieve an approximation ratio of $\frac{(e-1)(Z-1)}{2e^2 Z (kR)^{1/(Z-1)}}$ and $\frac{(e-1)(Z-1)}{2e(Z-1+ZR^{1/(Z-1)})}$, respectively, where Z (to be defined later) is the amount of resource compared to flow demand.

A. Placement Algorithm

In this subsection, we prove that function $R_4(S)$ is sequence submodular. Then, using the property of sequence submodularity, we propose a greedy algorithm for solving the placement subproblem. A function from sequences to real numbers, $f : \mathcal{V}^* \rightarrow \mathbb{R}$, is sequence-submodular if:

$$\begin{aligned} &\forall A_1, A_2 \in \mathcal{V}^*, \text{ such that } A_1 \preceq A_2, \ \forall v \in \mathcal{V}, \\ &f(A_1 \oplus (v)) - f(A_1) \geq f(A_2 \oplus (v)) - f(A_2). \end{aligned} \tag{8}$$

Also, function f is sequence-monotone if:

$$\forall A_1, A_2 \in \mathcal{V}^*, \text{ such that } A_1 \preceq A_2, \text{then } f(A_2) \geq f(A_1). \tag{9}$$

Next, we have the following lemma:

Lemma 3. *The function $R_4(S)$ is sequence-submodular and sequence-monotone.*

Proof. First, we show that function $R_4(S)$ is sequence-monotone (i.e., satisfies Eq. (9)). Since $A_1 \preceq A_2$, then according to Corollary 1, the assigned traffic to nodes A_1 will be the same for the two sequences A_1 and A_2. Adding additional nodes to sequence A_1 will not affect what has been already assigned to nodes A_1, and the minimum that can be assigned to any node is zero. So, Eq. (9) is satisfied.

Next, we show that function $R_4(S)$ is sequence-submodular (i.e., satisfies Eq. (8)). We denote the additional traffic that will be assigned when adding sequence A' to sequence A as $R_4(A'|A) \triangleq R_4(A \oplus A') - R_4(A)$. Also, remember that we use $\bar{\lambda}_v^A$ to denote the amount of traffic assigned to node v when solving $R_4(A)$. For Eq. (8) to be satisfied, it is sufficient to show that for any $v' \in \mathcal{V}$, $R_4((v')|A_1) \geq R_4((v')|A_2)$. This can be shown as follows:

$$\begin{aligned} R_4((v')|A_2) &= R_4(A_2 \oplus (v')) - R_4(A_2) \\ &\overset{\text{(b)}}{=} \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_2 \oplus (v')} + \sum_{i=|A_1|+1}^{|A_2|} \bar{\lambda}_{v_i}^{A_2 \oplus (v')} + \bar{\lambda}_{v'}^{A_2 \oplus (v')} - \\ &\quad \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_2} + \sum_{i=|A_1|+1}^{|A_2|} \bar{\lambda}_{v_i}^{A_2} \\ &\overset{\text{(c)}}{=} \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_2 \oplus (v')} + \bar{\lambda}_{v'}^{A_2 \oplus (v')} - \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_2} \\ &\overset{\text{(d)}}{\leq} \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_1 \oplus (v')} + \bar{\lambda}_{v'}^{A_1 \oplus (v')} - \sum_{i=1}^{|A_1|} \bar{\lambda}_{v_i}^{A_1} \\ &= R_4(A_1 \oplus (v')) - R_4(A_1) \\ &= R_4((v')|A_1), \end{aligned} \tag{10}$$

where (b) holds because $A_1 \preceq A_2$; (c) holds because nodes A_2 in sequence $A_2 \oplus (v')$ and sequence A_2 will be assigned the same amount of traffic (Corollary 1). Next, we show that (d) holds. Since $A_1 \preceq A_1 \oplus (v') \preceq A_2 \oplus (v')$, then, based on Corollary 1, the same amount of traffic will be assigned to nodes A_1 when solving $R_4(A_1 \oplus (v'))$ and $R_4(A_2 \oplus (v'))$. So, we only need to show that $\bar{\lambda}_{v'}^{A_2 \oplus (v')} \leq \bar{\lambda}_{v'}^{A_1 \oplus (v')}$. Based on the formulation in problem (2), the solution of $R_4((v')|A_2)$ satisfies all the constraints of nodes A_1, so $\bar{\lambda}_{v'}^{A_2 \oplus (v')}$ should be a feasible solution to node v' when solving $R_4(A_1 \oplus (v'))$. ☐

Because of this useful sequence submodular property, problem $(Q4)$ can be approximately solved using an efficient greedy algorithm. In this case, we can use a simple *Sequence Submodular Greedy* (SSG) algorithm to approximately solve problem $(Q4)$. In the SSG algorithm, we start with an empty solution of VNF-nodes S; in each iteration, we add a node that has the maximum marginal contribution to S, i.e., a node that leads to the largest increase in the value of the objective function $R_4(S)$. We repeat the above procedure until k VNF-nodes have been selected. The overall algorithm is presented in Algorithm 3. We state the performance of the SSG algorithm in the following lemma.

Algorithm 3 Sequence Submodular Greedy (SSG) algorithms

Input: set of nodes $\mathcal{V}$, set of flows $\mathcal{F}$, amount of resources, flow rates, flows demand, and $k = \lfloor B/b \rfloor$.
Output: sequence of VNF-nodes S.

```
Initialize: S = ()
while there is a node v such that |S ⊕ (v)| ≤ k do
    Pick a node v ∈ V and not in S that maximizes R_4(S ⊕ (v))
    S ← S ⊕ (v)
end while
```

Lemma 4. *The SSG algorithm achieves an approximation ratio of* $(1 - 1/e)$*, i.e.,* $R_4(S) \geq (1 - 1/e)OPT(Q4)$*.*

Proof. For an objective function that is sequence-submodular and sequence-monotone, it has been shown in [8, Theorem 3] that the greedy algorithm achieves an approximation ratio of $(1 - 1/e)$. We have shown in Lemma 3 that the objective function of problem $(Q4)$ is sequence-submodular and sequence-monotone, so the result of the lemma holds. □

B. Resource Allocation Algorithm

While solving the placement subproblem $(Q4)$, the resource allocation is achieved by using Algorithm 1, which allows partially processed flows to be counted. However, problem $(P1)$ requires flows to be fully processed. Therefore, we present two resource allocation algorithms that modify the resource allocation of the selected VNF-nodes while guaranteeing certain approximation ratios. Both algorithms are based on the primal-dual technique [26]. We describe each of the algorithms in the following.

We first provide a formulation of the optimal fractional resource allocation of sequence S, which allows partially processed flows. Based on the dual of this formulation, we will present the two resource allocation algorithms. We define $\delta_f^r \triangleq \sum_{\phi \in \Phi_f} \beta_\phi^r$ to be the total amount of resource r needed to process a unit of flow f by a set of network functions Φ_f. We define the maximum demand across all flows as $d_{\max} \triangleq \max_{f \in \mathcal{F}, r \in \mathcal{R}} \delta_f^r \lambda_f$. Then, for each flow f we define the normalized total demand of resource r as $d_f^r \triangleq \delta_f^r \lambda_f / d_{\max}$. In addition, for each VNF-node v, we define the normalized total amount of resource r as $\bar{c}_v^r \triangleq c_v^r / d_{\max}$. Finally, we define $Z \triangleq \min_{v \in \mathcal{S}, r \in \mathcal{R}} \bar{c}_v^r$ as a measure of the available resource compared to flow demand (we call it resource stretch). Since each VNF-node in sequence S is unique, then we can express sequence S as a set of VNF-nodes $\mathcal{S}$ and continue the discussion on the resource allocation of VNF-nodes $\mathcal{S}$. We use x_f^v to denote the portion of flow f that is assigned to VNF-node v and $\mathcal{S}_f \triangleq \mathcal{S} \cap \mathcal{V}_f$ to denote the set of VNF-nodes along the path of flow f that are in VNF-nodes $\mathcal{S}$. The optimal fractional resource allocation of VNF-nodes $\mathcal{S}$ can be formulated as:

$$
\begin{aligned}
&\max_{x_f^v} \sum_{f \in \mathcal{F}} \lambda_f \sum_{v \in \mathcal{S}_f} x_f^v \\
&\text{subject to} \\
&\sum_{f \in \mathcal{F}} d_f^r x_f^v \leq \bar{c}_v^r, \ \forall r \in \mathcal{R} \ \text{ and } v \in \mathcal{S}, \\
&\sum_{v \in \mathcal{S}_f} x_f^v \leq 1, \ \forall f \in \mathcal{F}, \\
&x_f^v \geq 0, \ \forall f \in \mathcal{F} \text{ and } v \in \mathcal{S}.
\end{aligned}
\tag{11}
$$

The corresponding dual linear program is

$$
\begin{aligned}
&\min_{y_v^r, z_f} \sum_{v \in \mathcal{S}} \sum_{r \in \mathcal{R}} \bar{c}_v^r y_v^r + \sum_{f \in \mathcal{S}_f} z_f \\
&\text{subject to} \\
&z_f + \sum_{r \in \mathcal{R}} d_f^r y_v^r \geq \lambda_f, \ \forall f \in \mathcal{F} \ \text{ and } v \in \mathcal{S}_f, \\
&y_v^r, z_f \geq 0, \ \forall v \in \mathcal{V} \ \text{ and } r \in \mathcal{R} \ \text{ and } f \in \mathcal{F}.
\end{aligned}
\tag{12}
$$

1) Primal-dual-based Resource Allocation (PRA): For the VNF-nodes $\mathcal{S}$ that are selected by the greedy algorithm, we modify their resource allocation to guarantee fully processed flows. We propose a primal-dual-based resource allocation algorithm, which is adapted from a multi-commodity routing algorithm proposed in [26] and based on the dual formulation (12). The main idea is to view the dual variable y_v^r as a price of resource r at VNF-node v. The algorithm chooses a VNF-node v_f with the minimum total cost for each flow. Then, it picks a flow that maximizes the relative value (i.e., λ_f) compared to the weighted cost and assigns that flow. Then, the price of each resource of the selected VNF-node is updated accordingly. The update of the price y_v^r is designed in a way such that if the limited resource is violated, then the stopping condition is satisfied from the previous iteration. The algorithm stops when all flows are assigned or when $\sum_{v \in \mathcal{S}} \sum_{r \in \mathcal{R}} \bar{c}_v^r y_v^r \geq e^{Z-1} R |\mathcal{S}|$. The update of price y_v^r is also implemented in a way such that it maintains the value of the dual problem within a range of the value of the primal problem. Then, by weak duality, this establishes the approximation ratio of the primal-dual algorithm.

We use $\pi_{\text{PRA}}^{\mathcal{S}}$ to denote the total traffic assigned to VNF-nodes $\mathcal{S}$ by the PRA algorithm. The approximation ratio of the PRA algorithm with respect to function $R_4(S)$ is stated in the following Lemma.

Lemma 5. *The approximation ratio of the PRA algorithm is* $\pi_{PRA}^{\mathcal{S}} \geq \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S)$.

Proof. We use OPT((11)) to denote the optimal value of the primal problem (11). The proof follows from the following:

$$
\begin{aligned}
\pi_{\text{PRA}}^{\mathcal{S}} &\overset{\text{(a)}}{\geq} \frac{Z-1}{eZ(|\mathcal{S}|R)^{1/(Z-1)}} \text{OPT}((11)) \\
&\overset{\text{(b)}}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} \text{OPT}((11)) \\
&\overset{\text{(c)}}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S).
\end{aligned}
\tag{13}
$$

Algorithm 4 Primal-dual-based resource allocation (PRA)

Input: VNF-nodes $\mathcal{S}$, set of flows $\mathcal{F}$, normalized resources $\bar{c}_v^r$, flow rates λ_f, normalized flow demands d_f^r.
Initialization: $y_v^r = 1/\bar{c}_v^r,\ \forall r \in \mathcal{R}, v \in \mathcal{S},\ \lambda_f^v = 0, \forall f \in \mathcal{F}$ and $v \in \mathcal{S}$
Output: $\boldsymbol{\lambda}$
repeat
  for $f \in \mathcal{F}$ **do**
    $v_f = \arg\min_{v \in \mathcal{S}_f}\{\sum_{r \in R} y_v^r\}$;
  end for
  $f' = \arg\max_{f \in \mathcal{F}}\{\frac{\lambda_f}{\sum_{r \in \mathcal{R}} d_f^r y_{v_f}^r}\}$;
  $\lambda_{f'}^{v_{f'}} = \lambda_{f'}$;
  $\mathcal{F} = \mathcal{F} \setminus \{f'\}$;
  update $y_{v_{f'}}^r = y_{v_{f'}}^r (e^{Z-1} R|\mathcal{S}|)^{d_{f'}^r/(\bar{c}_{v_{f'}}^r - 1)},\ \forall r \in \mathcal{R}$;
until $\sum_{v \in \mathcal{S}} \sum_{r \in \mathcal{R}} \bar{c}_v^r y_v^r \geq e^{Z-1} R|\mathcal{S}|$ or $\mathcal{F} = \emptyset$;

The primal-dual algorithm has been shown to achieve the approximation ratio in (a) with respect to any fractional solution [26, Lemma 5.7, Theorem 5.1]; (b) follows because $k \geq |\mathcal{S}|$; (c) follows from the fact that the value $R_4(S)$ is upper bounded by OPT((11)) □

When Z goes to infinity, then the algorithm has an approximation ratio of $1/e$. The time complexity of the PRA algorithm is $O(|\mathcal{S}|F^2)$.

2) Node-based Resource Allocation (NRA): The approximation ratio of the PRA algorithm depends on two parameters: the budget k and the resource stretch Z. If k is large and Z is small, then the approximation ratio of the PRA algorithm becomes small. However, if Z is large enough, then it will offset the effect of large k. Therefore, we design another algorithm, node-based resource allocation algorithm (NRA), which removes the dependence on k but adds a constant factor to the approximation ratio. The main idea of the NRA algorithm is to make the resource allocation of each VNF-node separately based on any order. For each VNF-node in $\mathcal{S}$, its resources are allocated using the primal-dual technique by considering the remaining unassigned flows. The detail of the NRA algorithm is presented in Algorithm 5. Similar to the PRA algorithm, we view the dual variable y_v^r as a price for each resource. The difference here is that we consider each VNF-node separately and try to assign flows with the largest ratio of the rate λ_f compared to the weighted demand $\sum_{r \in \mathcal{R}} d_f^r y_v^r$.

We use $\pi_{\text{NRA}}^{\mathcal{S}}$ to denote the total traffic assigned to VNF-nodes $\mathcal{S}$ by the NRA algorithm. We state the approximation ratio of the NRA algorithm in the following lemma.

Lemma 6. *The approximation ratio of the NRA algorithm is* $\pi_{NRA}^{\mathcal{S}} \geq \frac{Z-1}{Z-1+eZR^{1/(Z-1)}} R_4(S)$.

Proof. First, we define additional notations. Let $\mathcal{F}' \subseteq \mathcal{F}$ denote the set of unassigned flows by the end of Algorithm 5

Algorithm 5 Greedy Resource Allocation (NRA)

Input: VNF-nodes $\mathcal{S}$, set of flows $\mathcal{F}$, normalized resources $\bar{c}_v^r$, flow rates λ_f, normalized flow demands d_f^r.
Initialization: $\lambda_f^v = 0, \forall f \in \mathcal{F}$ and $v \in \mathcal{S}$
Output: $\boldsymbol{\lambda}$
for each VNF-node $vin\mathcal{S}$ **do**
  Initialization: $y_v^r = 1/\bar{c}_v^r,\ \forall r \in \mathcal{R}$
  repeat
    $f' = \arg\max_{f \in \mathcal{F}}\{\frac{\lambda_f}{\sum_{r \in \mathcal{R}} d_f^r y_v^r}\}$;
    $\lambda_{f'}^v = \lambda_{f'}$;
    $\mathcal{F} = \mathcal{F} \setminus \{f'\}$;
    update $y_v^r = y_v^r (e^{Z-1} R)^{d_f^r/(\bar{c}_v^r - 1)},\ \forall r \in \mathcal{R}$;
  until $\sum_{r \in \mathcal{R}} \bar{c}_v^r y_v^r \geq e^{Z-1} R$ or $\mathcal{F} = \emptyset$;
end for

and $\mathcal{F}_v$ denote the set of unassigned flows right before considering VNF-node v_i by Algorithm 5. We also use OPT($\mathcal{S}$) to denote the optimal resource allocation of VNF-nodes $\mathcal{S}$ and use OPT($\mathcal{S}|\bar{\mathcal{F}}$) to denote the optimal resource allocation of VNF-nodes $\mathcal{S}$ considering only a subset of flows $\bar{\mathcal{F}}$. The maximum traffic that can be assigned by any algorithm to VNF-nodes $\mathcal{S}$ has the following upper bound:

$$\begin{aligned} R_4(S) &\leq \text{OPT}(\mathcal{S}) \\ &\overset{(a)}{\leq} \sum_{f \in \mathcal{F} \setminus \mathcal{F}'} \lambda_f + \text{OPT}(\mathcal{S}|\mathcal{F}') \\ &\overset{(b)}{=} \pi_{\text{NRA}}^{\mathcal{S}} + \text{OPT}(\mathcal{S}|\mathcal{F}') \\ &\leq \pi_{\text{NRA}}^{\mathcal{S}} + \sum_{v \in \mathcal{S}} \text{OPT}(\{v\}|\mathcal{F}') \\ &\overset{(c)}{\leq} \pi_{\text{NRA}}^{\mathcal{S}} + \sum_{v \in \mathcal{S}} \text{OPT}(\{v\}|\mathcal{F}_v) \\ &\overset{(d)}{\leq} \pi_{\text{NRA}}^{\mathcal{S}} + \sum_{v \in \mathcal{S}} \frac{eZ}{Z-1} R^{1/(Z-1)}\ \pi_{\text{NRA}}^{\{v\}} \\ &\leq \pi_{\text{NRA}}^{\mathcal{S}} + \frac{eZ}{Z-1} R^{1/(Z-1)} \pi_{\text{NRA}}^{\mathcal{S}} \\ &= \frac{Z-1+eZR^{1/(Z-1)}}{Z-1} \pi_{\text{NRA}}^{\mathcal{S}}, \end{aligned} \tag{14}$$

where (a) holds because one possible upper bound is to consider what can be assigned from a subset of flows $\mathcal{F}'$ and add to it all other flows $\mathcal{F} \setminus \mathcal{F}'$; (b) holds because flows $\mathcal{F} \setminus \mathcal{F}'$ are all assigned by the NRA algorithm; (c) holds because $\mathcal{F}_v$ is a superset of $\mathcal{F}'$. For (d), the greedy algorithm for a single VNF-node achieves an approximation ratio of $\frac{eZ}{Z-1} R^{1/(Z-1)}$ with respect to any fractional solution [26, Lemma 5.7, Theorem 5.1], so (d) holds. □

When Z goes to infinity, then the approximation ratio is $1/(e+1)$. The time complexity of the NRA algorithm is $O(F^2)$.

C. Main Results

We state our main results in Theorems 1 and 2.

Theorem 1. *The SSG-PRA algorithm has an approximation ratio of* $\frac{(e-1)(Z-1)}{2e^2 Z(kR)^{1/(Z-1)}}$ *for problem* $(P1)$ *and becomes* $\frac{e-1}{2e^2}$ *when* $Z \to \infty$.

Proof. The SSG-PRA algorithm has two main components: 1) VNF-nodes placement and 2) resource allocation. We use $OPT(P)$ to denote the optimal value of any problem (P). We start with the result of the VNF-nodes placement using the SSG algorithm. For sequence S that is selected by the SSG algorithm, we have the following result:

$$\begin{aligned} R_4(S) &\overset{(a)}{\geq} (1-1/e)\text{OPT}(Q4) \\ &\overset{(b)}{\geq} \frac{1}{2}(1-1/e)\text{OPT}(Q3) \\ &\overset{(c)}{=} \frac{1}{2}(1-1/e)\text{OPT}(Q1) \\ &\overset{(d)}{\geq} \frac{1}{2}(1-1/e)\text{OPT}(P1), \end{aligned} \tag{15}$$

where (a) is due to Lemma 4, (b) holds from Lemma 2, (c) holds because an optimal resource allocation is assumed for the objective function of problem $(Q3)$, and (d) holds because problem $(Q1)$ is a relaxed version of problem $(P1)$.

The second component of the SSG-PRA algorithm is the resource allocation using the PRA algorithm for the sequence of VNF-nodes S selected by the SSG. We have the following result:

$$\begin{aligned} \pi^S_{\text{PRA}} &\overset{(a)}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S) \\ &\overset{(b)}{\geq} \frac{(e-1)(Z-1)}{2e^2 Z(kR)^{1/(Z-1)}} \text{OPT}(P1), \end{aligned} \tag{16}$$

where (a) comes from the approximation ratio of the PRA algorithm in Lemma 5, and (b) holds from Eq. (15). Therefore, the result of Theorem 1 follows. □

Theorem 2. *The SSG-NRA algorithm has an approximation ratio of* $\frac{(e-1)(Z-1)}{2e(Z-1+eZR^{1/(Z-1)})}$ *for problem* $(P1)$ *and becomes* $\frac{e-1}{2e^2+2e}$ *when* $Z \to \infty$.

Proof. The proof follows the same steps as the proof of Theorem 1. □

VII. Performance Evaluation

In this section, we complement our theoretical analysis of the proposed algorithms with a trace-driven simulation study. We compare the proposed algorithms with the optimal solution, obtained by solving the Integer Linear Program (ILP) formulation $(P1)$ using Gurobi solver (Gurobi 8.1.1). In addition, we conjecture that the objective function of placement subproblem $(Q3)$ is submodular. Therefore, we present the following two heuristics (SG-PRA algorithm and SG-NRA algorithm) based on this conjecture. In both heuristics, the placement is implemented in a similar way to that of the SSG algorithm, called Submodular Greedy (SG) algorithm [27]. Specifically, 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 $R_3(\mathcal{U})$. We repeat the above procedure until k VNF-nodes have been selected. Then, the resource allocation is implemented using the PRA (resp., NRA) algorithm for the SG-PRA (resp., SG-NRA) 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. Note that although we present the results of the optimal solution, the multi-VPRA problem is NP-hard in general (Lemma 1), and for some problem instances it may take a prohibitively large amount of time to finish solving the ILP formulation.

A. Evaluation Datasets

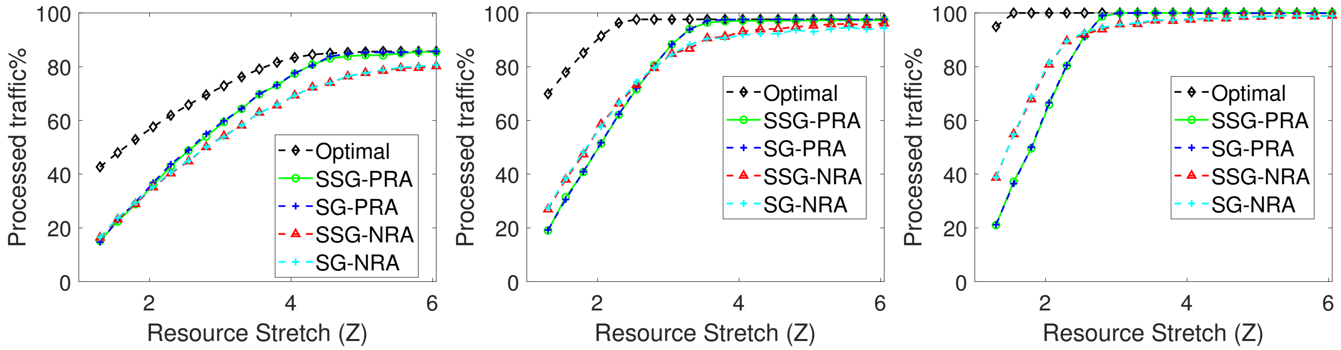
1) Abilene Dataset: We consider the Abilene dataset collected from an educational backbone network in North America [10]. The network consists of 12 nodes and 144 flows. Each flow rate was recorded every five minutes for 6 months. Also, OSPF weights were recorded, which allows us to compute the shortest 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. We consider two types of resources (i.e., $R = 2$), and the demand of each flow is chosen uniformly at random between 0 and 20 (i.e., $\delta^r_f \in [0, 20]$). The total available resource is set to the maximum total demand of flows d_{max} multiplied by a resource stretch $Z > 1$.

2) SNDlib Datasets: We also consider two other datasets from SNDlib [11]: Cost266 with 37 nodes and 1332 flows, and ta2 with 65 nodes and 1869 flows. For Cost266, the link's routing cost is available, so we use that to compute the shortest path of each flow. For ta2, we use hop-count-based shortest path. The setting of resources is the same as that of the Abilene dataset.

B. Evaluation Results

We start with the Abilene dataset, where we study the effect of having different values of resource stretch Z and budget B. Recall that Z is the ratio of the minimum available resource to the maximum flow demand. We consider a budget of 3, 6, and 10 VNF-nodes. The results are presented in Fig. 1. From the results, we make the following observations.

First, we can see that the simulation results for both the SSG-PRA and SSG-NRA algorithms agree with their approximation ratios presented in Theorems 1 and 2 in that when the budget k or Z is small, the SSG-NRA performs better and vice versa. Specifically, we start with Fig. 1(a) when the budget is 3. When the amount of resources is small or there are flows with huge demand (i.e., Z is small), the SSG-NRA algorithm is slightly better, but since the budget and number of resources (i.e., kR) is small anyway, it does not affect the performance of the SSG-PRA algorithm much. When Z becomes large (either by having a larger amount of resources or by having flows with a smaller demand to make $Z \geq 4$), the effect of the terms $(kR)^{1/(Z-1)}$ and $R^{1/(Z-1)}$ diminishes, but the effect of the constant term of the SSG-NRA algorithm remains as can be seen from the result when Z goes to infinity. That leads



(a) Budget = 3 VNF-nodes (b) Budget = 6 VNF-nodes (c) Budget = 10 VNF-nodes

Fig. 1: Evaluations based on Abilene dataset with different budget k and resource stretch Z

to a slightly worse performance for the SSG-NRA algorithm when Z is large. By doubling the budget to 6 VNF-nodes, we can see in Fig. 1(b) that the performance of the SSG-NRA algorithm is better than that of the SSG-PRA algorithm when Z is small (i.e., $Z \leq 2.5$). This is because when Z is small and kR is large, there is a high chance that the stopping condition of the PRA algorithm is satisfied early although some nodes still have a large amount of unused resources. In contrast, for the NRA algorithm, we consider nodes one by one, and if the stopping condition is satisfied early, it will only affect the node under consideration and the algorithm will continue allocating the resources of the other nodes. The same trend can also be seen in Fig. 1(c).

Second, although the SSG-NRA algorithm performs better when Z is small, sometimes it fails to reach the performance of the optimal solution even when Z is large (see Fig. 1(a)). Increasing the budget helps alleviating this problem with SSG-NRA algorithm, but still it needs at least twice the resource stretch Z needed by the SSG-PRA algorithm to reach a similar performance of the optimal solution (see Figures 1(b) and 1(c)). The proposed algorithms achieve at least $1/2$ of the optimal solution, which verifies our theoretical results.

Third, comparing the proposed algorithms with the two heuristics, we can see that the proposed algorithms perform almost the same as the heuristics. The proposed algorithms even perform better on multiple occasions as for the SSG-NRA algorithm. This seems to suggest that even if our conjecture of $R_3(\mathcal{U})$ being submodular is correct, the loss by considering the second-level relaxation is negligible. However, the second-level relaxation is important as it allows us to draw a connection to the sequence submodular theory and to establish the performance guarantee of the SSG algorithm.

Fourth, the results suggest that in order to gain the best performance in terms of total processed traffic, ISPs have two options: 1) either to scale resources vertically by provisioning more resources at each node (i.e., makes Z large); or 2) scale horizontally by deploying more VNF-nodes. Both of these options have shown promising performance as can be seen in Fig. 1.

Further, we also extend the evaluations to other datasets with a larger number of nodes and flows. We consider Cost266 dataset (for a network consisting of 37 nodes and 1332 flows) and ta2 dataset (for a network consisting of 65 nodes and 1869 flows). For these two datasets, the results appear to have a similar trend compared to that of Fig. 1. Due to space limitations, we provide these simulation results in our online technical report [28].

VIII. Conclusion

In this paper, we considered the problem of placement and resource allocation of VNF-nodes. We showed that considering the multi-dimensional setting along with the budget, limited resources, and fully flow processing constraints introduces several new challenges. However, through a two-level relaxation, we were able to develop an efficient placement algorithm. In addition, we utilized the primal-dual technique to design efficient resource allocation algorithms that properly handles the multi-dimensional setting. Although the second-level relaxation results in a smaller approximation ratio (a factor of 1/2), we showed through simulation that its impact of the empirical performance is negligible. Besides, the simulation results agree with the derived approximation ratio of both resource allocation algorithms. Specifically, the simulation showed that for a smaller resource stretch Z and larger number of nodes, the NRA algorithm works better; when Z becomes large enough, the PRA algorithm is better than the NRA algorithm and reaches the performance of the optimal solution earlier. In our future work, we will consider service function chaining, where the network functions required for each flow must be in a specific order.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, D. Telekom, and U. Michel, “Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action,” *in Proc. SDN OpenFlow World Congr., Darmstadt, Germany*, no. 1, pp. 1–16, 2012.

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

[3] 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.

[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 *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

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

[6] C. Chekuri and S. Khanna, "On multi-dimensional packing problems," in *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1999, pp. 185–194.

[7] Z. Zhang, E. K. Chong, A. Pezeshki, and W. Moran, "String submodular functions with curvature constraints," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 601–616, 2016.

[8] S. Alaei, A. Makhdoumi, and A. Malekian, "Maximizing sequence-submodular functions and its application to online advertising," *arXiv preprint arXiv:1009.4153*, 2010.

[9] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[10] "Abilene dataset, http://www.cs.utexas.edu/ yzhang/research/abilenetm/."

[11] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0âĂŤsurvivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[12] 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.

[13] 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.

[14] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," Ph.D. dissertation, Université Côte d'Azur, CNRS, I3S, France; Inria Sophia Antipolis, 2018.

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

[16] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2015, pp. 104–118.

[17] B. Ren, D. Guo, Y. Shen, G. Tang, and X. Lin, "Embedding service function tree with minimum cost for nfv-enabled multicast," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1085–1097, 2019.

[18] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *INFOCOM Conference on Computer Communications, IEEE*. IEEE, 2018.

[19] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation Algorithms for the NFV Service Distribution Problem," *IEEE INFOCOM, Atlanta, GA, May 2017*, pp. 1–9, 2017.

[20] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "Cluspr: Balancing multiple objectives at scale for nfv resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.

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

[22] H. Yu, J. Yang, C. Fung, R. Boutaba, and Y. Zhuang, "Ensc: Multi-resource hybrid scaling for elastic network service chain in clouds," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 34–41.

[23] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," *Theoretical Computer Science*, vol. 496, pp. 184–194, 2013.

[24] T. Wang, H. Xu, and F. Liu, "Multi-resource load balancing for virtual network functions," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1322–1332.

[25] S. Tschiatschek, A. Singla, and A. Krause, "Selecting sequences of items via submodular maximization," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] P. Briest, P. Krysta, and B. Vöcking, "Approximation techniques for utilitarian mechanism design," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1587–1622, 2011.

[27] 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.

[28] G. Sallam, Z. Zheng, and B. Ji, "Placement and allocation of virtual network functions: Multi-dimensional case," *arXiv preprint arXiv:1910.06299*, 2019.