

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 served in an NFV-enabled network. 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); 2) allocation of the VNF-nodes' resources to network flows. These two subproblems must be jointly considered to satisfy the objective of serving the maximum amount of traffic. This problem has been studied 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 consider the multi-dimensional setting, where flows must be processed by multiple network functions, which 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 subproblem a multi-dimensional generalization of the generalized assignment problem with assignment restrictions. To address these difficulties, we propose a novel two-level relaxation method that allows us to draw a connection to the sequence submodular theory and utilize the property of sequence submodularity along with the primal-dual technique to design two approximation algorithms. We further prove that the proposed algorithms have a non-trivial approximation ratio that depends on the number of VNF-nodes, resources, and a measure of the available resource compared to flow demand. Finally, we perform trace-driven simulations to show the effectiveness of the proposed algorithms.

1 INTRODUCTION

With the emergence of many new technologies and applications (such as autonomous vehicles, extended reality, and edge intelligence), the role of modern networks (beyond-5G and 6G) has evolved beyond providing basic connectivity services [2]. It has now become imperative to provide various types of network services (such as security, performance optimization, and value-added services) in modern networks. Realizing such network services usually requires

a new design paradigm, called Network function virtualization (NFV), where network functions (e.g., firewall, intrusion detection, and load balancer) that traditionally run in dedicated hardware are now replaced by software modules hosted on general purpose commodity servers [3]. 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 major Internet service providers (ISPs) to consider this new architecture, and some of them have already started the transition to NFV [4].

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, and I/O). In addition, flows generally require all of their traffic be fully processed by such functions to satisfy certain quality of services [5]. From ISPs' perspective, transitioning to NFV usually happens in multiple stages for several reasons, including budget limitations and the desire to utilize the already provisioned hardware. Considering both 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 consider a limited budget (e.g., [6]) or relaxes the resources constraint (e.g., [5]). In [7], both the budget and resources constraints are considered, along with the requirement that flows must be fully processed. 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; 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 [7] that not only is this problem NP-hard, but it does not possess a useful property, called submodularity, which oftentimes leads to efficient solutions. In this work, we take one step further and extend the basic-VPRA prob-

This work was supported in part by the NSF under Grants CNS-1651947. A preliminary version of this work was presented at IEEE ICNP 2019 [1].

Gamal Sallam (tug43066@temple.edu) is 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, and Bo Ji (boji@vt.edu) is with the Department of Computer Science, Virginia Tech, Blacksburg, VA. Bo Ji is the corresponding author.

lem to 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 design strategies and analytical techniques. Specifically, we decompose the original problem into two subproblems: placement and resource allocation. We show that the placement subproblem even without the requirement that flows must be fully processed is NP-hard. Moreover, the resource allocation subproblem is a multi-dimensional generalization of the generalized assignment problem with assignment restrictions, which is also NP-hard [8]. To address the placement subproblem, we introduce a novel two-level relaxation method that allows us to draw a connection to the sequence submodular (also called string submodular) theory [9], [10] and design an efficient placement algorithm. Note that sequence submodularity is a generalization of submodularity. Like submodular functions, sequence submodular functions also exhibit the diminishing returns property, meaning that the value of adding an item to a sequence decreases as the sequence expands. For sequence functions, forward (resp., backward) monotonicity means the value of the function increases when an item is added to the end (resp., beginning) of the sequence. We provide formal definitions of these properties in Section 5.1. For the resource allocation subproblem, we utilize the primal-dual technique [11] 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. In order to leverage the property of sequence submodularity, we generalize the concept of backward-monotone to approximate backward-monotone, extend the known results for backward-monotone to this generalized version, and utilize this new property to develop an efficient algorithm for the placement of VNF-nodes.
- Third, we utilize the primal-dual technique to design two efficient resource allocation algorithms. Moreover, we show that by combining the proposed placement algorithm and the two resource allocation algorithms, we can achieve an approximation ratio of $\frac{(e-1)(Z-1)}{4e^2 Z(kR)^{1/(Z-1)}}$ and $\frac{(e-1)(Z-1)}{4e(Z-1+eZR^{1/(Z-1)})}$ for the original non-relaxed multi-VPRA problem, respectively, where k (resp. R)

Symbol	Definition
\mathcal{V}	The set of nodes in the network
\mathcal{E}	The set of edges connecting the nodes in the network
\mathcal{F}	The set of flows
λ_f	The traffic rate of flow f
λ	The flow rate vector
\mathcal{V}_f	The set of nodes along the path of flow f
$\mathcal{F}_{\mathcal{U}}$	The set of all flows whose path has at least one node in a subset of nodes \mathcal{U}
Φ	The set of network functions
Φ_f	The set of network functions required by flow f
$\mathcal{F}(\phi)$	The set of flows that require network function ϕ
\mathcal{R}	The set of resource types
c_v^r	The total amount of resource r available at node v
β_ϕ^r	The amount of resource r needed for function ϕ to process one unit of a network flow
x_f^v	The portion of flow f assigned to VNF-node v
$\mathbf{X}(\mathcal{U})$	The flow assignment matrix of a set of nodes \mathcal{U}

TABLE 1: Summary of notations

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 ratios become constants: $\frac{e-1}{4e^2}$ and $\frac{e-1}{4e^2+4e}$, respectively.

- Finally, we conduct trace-driven simulations using Abilene dataset [12] as well as datasets from SNDlib [13] to evaluate the performance of the proposed algorithms.

2 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 $f \in \mathcal{F}$. Let $\lambda \triangleq [\lambda_{f_1}, \dots, \lambda_{f_F}]$ be the flow rate vector. As we mention earlier, the transition to NFV happens in two phases: the planning phase and the production phase. In this work we focus on the planning phase and assume that 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. We assume that flows can get processed by all required network functions at the same VNF-node. This has the potential of reducing the overhead of maintaining flow state across VNF-nodes and can be realized by the NFV architecture [3], which allows hosting different types of network functions at the same VNF-node. 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, CPU, and I/O), 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 x_f^v to denote the

portion of flow f that is assigned to VNF-node v and use $\mathbf{X}(\mathcal{V}) \in \mathcal{R}^{F \times V}$ to denote the assignment matrix. In Table 1, we summarize the main notations that will be used in the problem formulation.

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., $\sum_{v \in \mathcal{V}_f \cap \mathcal{U}} x_f^v \geq \lambda_f$, 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:

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

where $\mathbf{1}_{\{\cdot\}}$ 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:

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

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.,

$$|\mathcal{U}| \leq k. \quad (3)$$

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 ($J_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$). The problem can be formulated as follows:

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

3 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.

3.1 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 $J_2(\mathbf{X}(\mathcal{U}))$ denote the total amount of fully processed traffic under flow assignment $\mathbf{X}(\mathcal{U})$. Then, the resource allocation subproblem for a given set of VNF-nodes \mathcal{U} can be formulated as follows:

$$\begin{aligned} & \underset{\mathbf{X}(\mathcal{U}): (2) \text{ is satisfied}}{\text{maximize}} && J_2(\mathbf{X}(\mathcal{U})). \end{aligned} \quad (P2)$$

Let $J_3(\mathcal{U}) \triangleq \max_{\mathbf{X}(\mathcal{U}): (2) \text{ is satisfied}} J_2(\mathbf{X}(\mathcal{U}))$ 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 follows:

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

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

3.2 Hardness

In [7, 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 [7, 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 [7] employs a simple relaxation method that allows any portion of a partially processed flow to be counted in the objective function. That is, the objective function in Eq. (1) becomes $\sum_{f \in \mathcal{F}} \sum_{v \in \mathcal{V}_f \cap \mathcal{U}} x_f^v$. Such a simple 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 why the same framework and algorithms cannot be directly applied to solve the multi-VPRA problem we consider.

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 [7], 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 the sequence submodular theory and design an efficient placement algorithm. 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.

4 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 $R_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$ to denote the relaxed objective function (defined in Eq. (4)). In the second-level, instead of evaluating function $R_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$ 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.

4.1 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 $J_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$ can be expressed as follows:

$$R_1(\mathcal{U}, \mathbf{X}(\mathcal{U})) \triangleq \sum_{f \in \mathcal{F}} \sum_{v \in \mathcal{V}_f \cap \mathcal{U}} x_f^v. \quad (4)$$

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

$$\sum_{v \in \mathcal{U}} x_f^v \leq \lambda_f, \quad \forall f \in \mathcal{F}. \quad (5)$$

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

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

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 $\mathcal{X}^{\mathcal{U}}$ be the set of all flow assignment matrices $\mathbf{X}(\mathcal{U})$ that satisfy the resources constraint (2) and the flow rate constraint (5), and let $R_2(\mathbf{X}(\mathcal{U}))$ be the total processed traffic. Then, the resource allocation subproblem for a given set of VNF-nodes \mathcal{U} can be formulated as

$$\underset{\mathbf{X}(\mathcal{U}) \in \mathcal{X}^{\mathcal{U}}}{\text{maximize}} \quad R_2(\mathbf{X}(\mathcal{U})). \quad (Q2)$$

Now, let $R_3(\mathcal{U}) \triangleq \max_{\mathbf{X}(\mathcal{U}) \in \mathcal{X}^{\mathcal{U}}} R_2(\mathbf{X}(\mathcal{U}))$ 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

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

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.

4.2 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). By doing this, we make a connection to the sequence submodular theory, which enables us to design an efficient placement algorithm with provable performance guarantee. First, we give some additional notations. Let (v_1, v_2, \dots, v_m) be a sequence of nodes selected over m steps, where $v_i \in \mathcal{V}$ is selected in the i -th step for $i = 1, \dots, m$. Let the set of all possible sequences of nodes be $\mathcal{H} \triangleq \{(v_1, v_2, \dots, v_m) \mid m \in \mathbb{N} \cup \{0\}, v_i \in \mathcal{V}\}$; when $m = 0$, we have an empty sequence. For two sequences $S_1 = (v_1, v_2, \dots, v_{m_1})$ and $S_2 = (u_1, u_2, \dots, u_{m_2})$ in \mathcal{H} , we define a concatenation of S_1 and S_2 as

$$S_1 \oplus S_2 \triangleq (v_1, v_2, \dots, v_{m_1}, u_1, u_2, \dots, u_{m_2}).$$

We say that $S_1 \preceq S_2$ if S_2 can be rewritten as $S_1 \oplus S_3$ for some $S_3 \in \mathcal{H}$. For sequence S , we use $\mathcal{V}(S)$ to denote the set of nodes in sequence S . By slightly abusing the notation, we use $|S|$ to denote the number of elements in sequence S . In addition, we often use v to denote a singleton sequence (v) when there is no confusion

In the following, unless stated otherwise, we only consider sequences with unique nodes. For sequence S , let $y_i(S)$ denote the total flow assigned to the i -th node and let $\mathbf{y}(S)$ denote a given feasible resource allocation vector for sequence S . By slightly abusing the notation, we use $\mathbf{X}(S)$ to denote the flow assignment matrix of sequence S . Consider any node v_j in sequence S , with $j = 1, \dots, |S|$. Given a fixed resource allocation $\mathbf{y}(S)$, we define a fractional resource allocation of node v_j as the solution of the

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: $\hat{\mathbf{y}}(S)$ and $\hat{\mathbf{X}}(S)$
1: Initialize: $y_i(S) = 0$, $i = 1, \dots, |S|$
2: **for** $i = 1$ to $|S|$ **do**
3: $\mathbf{X}(S) \leftarrow$ solve Problem (6) for node v_i given $\mathbf{y}(S)$
4: Set $y_i(S) = \sum_{j=1}^F x_{f_j}^{v_i}$
5: **end for**
6: $\hat{\mathbf{X}}(S) = \mathbf{X}(S)$, $\hat{\mathbf{y}}(S) = \mathbf{y}(S)$

following problem:

$$\begin{aligned} & \underset{\mathbf{X}(S)}{\text{maximize}} && \sum_{f \in \mathcal{F}} x_f^{v_j} \\ & \text{subject to} && \sum_{v \in \mathcal{V}_f \cap \mathcal{V}(S)} x_f^v \leq \lambda_f, \quad \forall f \in \mathcal{F} \text{ and } v \in \mathcal{V}_f, \\ & && x_f^v = 0, \quad \forall f \in \mathcal{F} \text{ and } v \notin \mathcal{V}_f, \\ & && \sum_{\phi \in \Phi} \beta_\phi^r \sum_{f \in \mathcal{F}(\phi)} x_f^{v_i} \leq c_{v_i}^r, \quad \forall r \in \mathcal{R} \text{ and } i = 1, \dots, |S|, \\ & && \sum_{f \in \mathcal{F}} x_f^{v_i} = y_i(S), \quad i = 1, \dots, |S| \text{ and } i \neq j. \end{aligned} \quad (6)$$

In Problem (6), we want to maximize the total traffic that can be assigned to node v_j while satisfying the given resource allocation $\mathbf{y}(S)$ of all other nodes in S .

Now, consider sequence S . The resource allocation of nodes in S is presented in Algorithm 1. Algorithm 1 starts by initializing the total traffic assigned to each node to zero (i.e., set $y_i(S) = 0$, $i = 1, \dots, |S|$), and then iterates over nodes in sequence S according to their order. In iteration i , it computes the resource allocation of node v_i by solving Problem (6) given $\mathbf{y}(S)$, and then update $y_i(S)$ according to the obtained solution. We use $\hat{\mathbf{X}}(S)$ to denote the flow assignment matrix of sequence S at the end of Algorithm 1. Similarly, we use $\hat{\mathbf{y}}(S)$ to denote the resource allocation vector for sequence S at the end of Algorithm 1. For any sequence S , we define function $R_4(S)$ to be the total traffic assigned by Algorithm 1 for nodes in sequence S , i.e.,

$$R_4(S) \triangleq \sum_{i=1}^{|S|} \hat{y}_i(S). \quad (7)$$

Note that for sequences with repeated nodes, the value of $R_4(S)$ is the same as $R_4(\bar{S})$, where \bar{S} is obtained from sequence S by removing all the later appearances of the same node. Then, the relaxed version of Problem (Q3) becomes the following:

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

Next, we will show that function $R_4(S)$ is a 1/2-approximation of function $R_3(\mathcal{U})$ for any sequence S that is a permutation of nodes in set \mathcal{U} . This ensures that an optimal solution for Problem (Q4) is a 1/2-approximation 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).

First, we present Lemmas 2 and 3, which will be used in the proof of the approximation ratio of Algorithm 1 and in establishing that function R_4 is sequence-submodular in the next section.

Lemma 2. Consider S_1 , S_2 , and S_3 in \mathcal{H} , such that $S_3 \preceq S_1$ and $S_3 \preceq S_2$. Applying Algorithm 1 to S_1 and S_2 , respectively, yields $\hat{y}_i(S_1) = \hat{y}_i(S_2)$, for $i = 1, \dots, |S_3|$.

Proof. See Section 9.1. \square

Before we present Lemma 3, we define some additional notations. By slightly abusing the notation, we use $\hat{\mathbf{X}}(\mathcal{U})$ to denote an optimal flow assignment matrix of nodes \mathcal{U} after solving Problem (Q2). We use $\hat{x}_j(\mathcal{U})$ to denote the total traffic assigned from flow f_j to nodes \mathcal{U} , i.e.,

$$\hat{x}_j(\mathcal{U}) = \sum_{v \in \mathcal{U}} \hat{x}_{f_j}^v(\mathcal{U}), \quad (8)$$

and we define $\hat{\mathbf{x}}(\mathcal{U}) \triangleq [\hat{x}_1(\mathcal{U}), \dots, \hat{x}_F(\mathcal{U})]$. We can express $R_3(\mathcal{U})$ in terms of $\hat{\mathbf{x}}(\mathcal{U})$ as follows:

$$R_3(\mathcal{U}) = \sum_{j=1}^F \hat{x}_j(\mathcal{U}). \quad (9)$$

Similarly, given $\hat{\mathbf{X}}(S)$, which is the flow assignment matrix in sequence S at the end of Algorithm 1, we use $\hat{x}_j(S)$ to denote the total traffic assigned from flow f_j to nodes of sequence S , i.e.,

$$\hat{x}_j(S) = \sum_{i=1}^{|S|} \hat{x}_{f_j}^{v_i}(S), \quad (10)$$

and $\hat{\mathbf{x}}(S) \triangleq [\hat{x}_1(S), \dots, \hat{x}_F(S)]$. We can express $R_4(S)$ in terms of $\hat{\mathbf{x}}(S)$ as follows:

$$R_4(S) = \sum_{j=1}^F \hat{x}_j(S). \quad (11)$$

We use $R_3(\mathcal{U}|\mathbf{x})$ to denote the value of function $R_3(\mathcal{U})$ from a given flow rate vector \mathbf{x} ; We use $R_4(S|\mathbf{x})$ in a similar manner. Note that $R_3(\mathcal{U})$ is equivalent to $R_3(\mathcal{U}|\boldsymbol{\lambda})$; similarly, we have $R_4(S) = R_4(S|\boldsymbol{\lambda})$. In the sequel, we consider element-wise operations on vectors. For two flow rate vectors \mathbf{x}_1 and \mathbf{x}_2 such that $\mathbf{x}_1 \leq \mathbf{x}_2$, we have

$$R_3(\mathcal{U}|\mathbf{x}_1) \leq R_3(\mathcal{U}|\mathbf{x}_2), \quad (12)$$

where the inequality holds because any feasible solution to Problem (Q2) for node \mathcal{U} given \mathbf{x}_1 is also a feasible solution to Problem (Q2) for node \mathcal{U} given \mathbf{x}_2 . In addition, for any node $u \in \mathcal{V}$, we have $R_4(u) = R_3(\{u\})$ because when applying Algorithm 1 to a singleton sequence, the equality constraints of Problem (6) are irrelevant, which makes Problem (6) equivalent to Problem (Q2). As a result, we obtain the following:

$$R_4(u|\mathbf{x}_1) \leq R_4(u|\mathbf{x}_2). \quad (13)$$

Note that Eq. (13) does not hold for non-singleton sequence in general.

Next, we present Lemma 3.

Lemma 3. Consider S_1 and S_2 in \mathcal{H} such that $S_1 \preceq S_2$. For any node $u \notin \mathcal{V}(S_1)$, we have

$$R_4(u|\lambda - \hat{x}(S_2)) \leq R_4(S_1 \oplus u|\lambda) - R_4(S_1|\lambda). \quad (14)$$

Proof. See Section 9.2. \square

The approximation ratio of Algorithm 1 is stated in the following lemma.

Lemma 4. For a given set of nodes \mathcal{U} , let $\mathcal{P}(\mathcal{U})$ be the set of all permutations of nodes \mathcal{U} . For any $S \in \mathcal{P}(\mathcal{U})$, we have $\frac{1}{2}R_3(\mathcal{U}) \leq R_4(S) \leq R_3(\mathcal{U})$.

Proof. See Section 9.3. \square

5 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 forward-monotone, approximate backward-monotone, and sequence-submodular (to be defined in Section 5.1). 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 $\frac{(e-1)(Z-1)}{4e^2 Z(kR)^{1/(Z-1)}}$ and $\frac{(e-1)(Z-1)}{4e(Z-1+ZR^{1/(Z-1)})}$, respectively, where Z (to be defined in Section 5.3) is the amount of resource compared to flow demand.

5.1 Preliminary Results

In this subsection, we present results related to sequence submodular functions, which will be used to derive a placement algorithm for Problem (Q4). Note that the definitions and results presented in this subsection generalize to sequences with repeated nodes. We start with some definitions. A function from sequences to real numbers, $h : \mathcal{H} \rightarrow \mathbb{R}$, is sequence-submodular if

$$\forall S_1, S_2 \in \mathcal{H}, \text{ such that } S_1 \preceq S_2, \forall v \in \mathcal{V}, \quad h(S_1 \oplus v) - h(S_1) \geq h(S_2 \oplus v) - h(S_2). \quad (15)$$

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 $\mathbf{X}(\mathcal{U})$

- 1: **Relaxed Problem:** relax function $J_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$ to become $R_1(\mathcal{U}, \mathbf{X}(\mathcal{U}))$ (first-level relaxation), and relax function $R_3(\mathcal{U})$ to function $R_4(S)$ (second-level relaxation)
- 2: **Placement Subproblem:** solve Problem (Q4) using the sequence submodular greedy algorithm (Algorithm 3) to obtain S ; let $\mathcal{U} = \mathcal{V}(S)$
- 3: **Resource Allocation:** use either the PRA algorithm (Algorithm 4) or the NRA algorithm (Algorithm 5) to obtain resource allocation $\mathbf{X}(\mathcal{U})$ for nodes \mathcal{U}

Algorithm 3 Sequence Submodular Greedy (SSG) algorithm

Input: nodes \mathcal{V} , k
Initialization: $S = ()$
Output: S
1: **while** $|S| < k$ **do**
2: $S = S \oplus \arg \max_{v \in \mathcal{V}} (h(S \oplus v) - h(S))$
3: **end while**

Also, function h is forward-monotone if

$$\forall S_1, S_2 \in \mathcal{H}, h(S_1 \oplus S_2) \geq h(S_1), \quad (16)$$

and function h is backward-monotone if

$$\forall S_1, S_2 \in \mathcal{H}, h(S_1 \oplus S_2) \geq h(S_2). \quad (17)$$

Consider the problem of selecting a sequence S of length k that maximizes function $h(S)$, i.e.,

$$\max_{S \in \mathcal{H}: |S| \leq k} h(S). \quad (W)$$

Although Problem (W) is NP-hard, it has been shown in [14] that for function h that is forward-monotone, backward-monotone, and sequence-submodular, the *Sequence Submodular Greedy* (SSG) algorithm, presented in Algorithm 3, achieves an approximation of $(1 - 1/e)$. Algorithm 3 starts with an empty sequence S and greedily adds a node that has the largest incremental value to sequence S until $|S| = k$.

However, some functions may only satisfy an approximate version of the backward-monotone property (e.g., function $R_4(S)$ as shown in Lemma 5). Therefore, we generalize the backward-monotone property as follows: For $\alpha \in (0, 1]$, function h is α -backward-monotone if

$$\forall S_1, S_2 \in \mathcal{H}, h(S_1 \oplus S_2) \geq \alpha h(S_2). \quad (18)$$

Then, in the following theorem, we derive the approximation ratio of Algorithm 3 for function h that satisfies the α -backward-monotone property. This result will be used later to design an efficient algorithm for our VNF-node placement problem. In the sequel, for any Problem (P), we use $\text{OPT}(P)$ to denote its optimal value. Without loss of generality, we assume that the value of an empty sequence is zero.

Theorem 1. Suppose that sequence function h is forward-monotone, α -backward-monotone, and sequence-submodular.

Then, Algorithm 3 achieves an approximation ratio of $\alpha(1 - 1/e)$ for Problem (W), i.e., $h(S) \geq \alpha(1 - 1/e)OPT(W)$.

Proof. The proof follows a similar line of analysis as in [10]. See Section 9.4. \square

5.2 Placement Algorithm

In this subsection, we prove that function $R_4(S)$ is forward-monotone, $\frac{1}{2}$ -backward-monotone, and sequence-submodular. Then, using the property of sequence submodularity, we employ the SSG algorithm (Algorithm 3) for solving the placement subproblem. We start with the following lemma.

Lemma 5. *The function $R_4(S)$ is forward-monotone, $\frac{1}{2}$ -backward-monotone, and sequence-submodular.*

Proof. See Section 9.5. \square

We would like to point out that function R_4 is $\frac{1}{2}$ -backward-monotone and that the bound of $\frac{1}{2}$ is tight. We prove it through constructing a problem instance in Section 9.6.

Because of this useful property of sequence submodularity, Problem (Q4) can be approximately solved using the SSG algorithm (Algorithm 3). In the SSG algorithm, we start with an empty solution of VNF-nodes in 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. Note that if a node has been selected in a previous iteration, then its marginal contribution in any subsequent iteration will be zero. If at any iteration the marginal contribution of all nodes is zero, then we select a node that has not been selected before. In this way, we ensure that the selected sequence S has no repeated nodes. To solve Problem (Q3), we need a set of nodes rather than a sequence, but in order to use Algorithm 3 and take advantage of its approximation ratio, we select a sequence of unique nodes, which can be converted to a set. We state the performance of the SSG algorithm for Problem (Q4) in the following theorem.

Theorem 2. *The SSG algorithm achieves an approximation ratio of $\frac{1}{2}(1 - 1/e)$ for Problem (Q4), i.e., $R_4(S) \geq \frac{1}{2}(1 - 1/e)OPT(Q4)$.*

Proof. The result follows immediately from Theorem 1 and Lemma 5 by plugging $\alpha = \frac{1}{2}$ into Theorem 1. \square

5.3 Resource Allocation Algorithms

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 [15]. We describe each of the algorithms in the following.

For the selected sequence S of VNF-nodes, let $\mathcal{U} = \mathcal{V}(S)$. We first provide a formulation of the optimal fractional

resource allocation of VNF-nodes \mathcal{U} , 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 $d_{\max}^r \triangleq \max_{f \in \mathcal{F}, r \in \mathcal{R}} \delta_f^r \lambda_f$. Then, for each flow f we define the normalized total demand of resource r as $d_f^r \triangleq \delta_f^r \lambda_f / d_{\max}^r$. In addition, for each VNF-node v , we define the normalized total amount of resource r as $\bar{c}_v^r \triangleq c_v^r / d_{\max}^r$. 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). We use a_f^v to denote the fraction of flow f that is assigned to VNF-node v . The optimal fractional resource allocation of VNF-nodes \mathcal{U} can be formulated as:

$$\begin{aligned} & \max_{a_f^v} \sum_{f \in \mathcal{F}} \lambda_f \sum_{v \in \mathcal{U} \cap \mathcal{V}_f} a_f^v \\ & \text{subject to} \\ & \sum_{f \in \mathcal{F}} d_f^r a_f^v \leq \bar{c}_v^r, \quad \forall r \in \mathcal{R} \text{ and } v \in \mathcal{U}, \\ & \sum_{v \in \mathcal{S}_f} a_f^v \leq 1, \quad \forall f \in \mathcal{F}, \\ & a_f^v \geq 0, \quad \forall f \in \mathcal{F} \text{ and } v \in \mathcal{U}. \end{aligned} \quad (19)$$

The corresponding dual linear program is

$$\begin{aligned} & \min_{b_v^r, z_f} \sum_{v \in \mathcal{U}} \sum_{r \in \mathcal{R}} \bar{c}_v^r b_v^r + \sum_{f \in \mathcal{U} \cap \mathcal{V}_f} z_f \\ & \text{subject to} \\ & z_f + \sum_{r \in \mathcal{R}} d_f^r b_v^r \geq \lambda_f, \quad \forall f \in \mathcal{F} \text{ and } v \in \mathcal{U} \cap \mathcal{V}_f, \\ & b_v^r, z_f \geq 0, \quad \forall v \in \mathcal{V}, r \in \mathcal{R} \text{ and } f \in \mathcal{F}. \end{aligned} \quad (20)$$

5.3.1 Primal-dual-based Resource Allocation (PRA)

For the VNF-nodes \mathcal{U} that are selected by the SSG 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 [15] and based on the dual formulation (20). The main idea is to view the dual variable b_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 of λ_f compared to the weighted cost and assigns that flow to the associated node. Then the price of each resource of the selected VNF-node is updated accordingly. The update of the price b_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 $\sum_{v \in \mathcal{U}} \sum_{r \in \mathcal{R}} \bar{c}_v^r b_v^r \geq e^{Z-1} R |\mathcal{U}|$. The update of price b_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 $\pi_{\text{PRA}}^{\mathcal{U}}$ to denote the total traffic assigned to VNF-nodes \mathcal{U} by the PRA algorithm. The approximation ratio of the PRA algorithm with respect to function $R_4(S)$ is stated in the following Lemma.

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

```

1: Input: VNF-nodes  $\mathcal{U}$ , set of flows  $\mathcal{F}$ , normalized re-
   sources  $\bar{c}_v^r$ , flow rates  $\lambda_f$ , normalized flow demands  $d_f^r$ .
2: Initialization:  $b_v^r = 1/\bar{c}_v^r$ ,  $\forall r \in \mathcal{R}, v \in \mathcal{U}$ ,  $x_f^v = 0, \forall f \in \mathcal{F}$  and  $v \in \mathcal{U}$ 
3: Output:  $\mathbf{X}(\mathcal{U})$ 
4: repeat
5:   for  $f \in \mathcal{F}$  do
6:      $v_f = \arg \min_{v \in \mathcal{U} \cap \mathcal{V}_f} \{\sum_{r \in \mathcal{R}} b_v^r\}$ ;
7:   end for
8:    $f' = \arg \max_{f \in \mathcal{F}} \{\frac{\lambda_f}{\sum_{r \in \mathcal{R}} d_f^r b_{v_f}^r}\}$ ;
9:    $x_{f'}^{v_{f'}} = \lambda_{f'}$ ;
10:   $\mathcal{F} = \mathcal{F} \setminus \{f'\}$ ;
11:  update  $b_{v_{f'}}^r = b_{v_{f'}}^r (e^{Z-1} R |\mathcal{U}|)^{d_{f'}^r / (\bar{c}_{v_{f'}}^r - 1)}$ ,  $\forall r \in \mathcal{R}$ ;
12: until  $\sum_{v \in \mathcal{U}} \sum_{r \in \mathcal{R}} \bar{c}_v^r b_v^r \geq e^{Z-1} R |\mathcal{U}|$  or  $\mathcal{F} = \emptyset$ ;

```

Lemma 6. The approximation ratio of the PRA algorithm is $\pi_{PRA}^{\mathcal{U}} \geq \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S)$.

Proof. Recall that we use OPT(19) to denote the optimal value of Problem (19). The proof follows from the following:

$$\begin{aligned}
\pi_{PRA}^{\mathcal{U}} &\stackrel{(a)}{\geq} \frac{Z-1}{eZ(|\mathcal{U}|R)^{1/(Z-1)}} \text{OPT}(19) \\
&\stackrel{(b)}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} \text{OPT}(19) \\
&\stackrel{(c)}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S).
\end{aligned} \tag{21}$$

The primal-dual algorithm has been shown to achieve the approximation ratio in (a) with respect to any fractional solution [15, Lemma 5.7, Theorem 5.1]; (b) follows because $k \geq |\mathcal{U}|$; (c) follows from the fact that the value $R_4(S)$ is upper bounded by OPT(19). \square

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

5.3.2 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{U} , 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 b_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 $\sum_{r \in \mathcal{R}} d_f^r b_v^r$.

Algorithm 5 Node-based Resource Allocation (NRA)

```

1: Input: VNF-nodes  $\mathcal{U}$ , set of flows  $\mathcal{F}$ , normalized re-
   sources  $\bar{c}_v^r$ , flow rates  $\lambda_f$ , normalized flow demands  $d_f^r$ .
2: Initialization:  $x_f^v = 0, \forall f \in \mathcal{F}, v \in \mathcal{U}$ 
3: Output:  $\mathbf{X}(\mathcal{U})$ 
4: for each VNF-node  $v \in \mathcal{U}$  do
5:   Initialization:  $b_v^r = 1/\bar{c}_v^r$ ,  $\forall r \in \mathcal{R}$ 
6:   repeat
7:      $f' = \arg \max_{f \in \mathcal{F}} \{\frac{\lambda_f}{\sum_{r \in \mathcal{R}} d_f^r b_v^r}\}$ ;
8:      $x_{f'}^v = \lambda_{f'}$ ;
9:      $\mathcal{F} = \mathcal{F} \setminus \{f'\}$ ;
10:    update  $b_v^r = b_v^r (e^{Z-1} R)^{d_{f'}^r / (\bar{c}_v^r - 1)}$ ,  $\forall r \in \mathcal{R}$ ;
11:   until  $\sum_{r \in \mathcal{R}} \bar{c}_v^r b_v^r \geq e^{Z-1} R$  or  $\mathcal{F} = \emptyset$ ;
12: end for

```

We use $\pi_{NRA}^{\{v\}}$ to denote the total traffic assigned to VNF-node v by the NRA algorithm and define $\pi_{NRA}^{\mathcal{U}} \triangleq \sum_{v \in \mathcal{U}} \pi_{NRA}^{\{v\}}$. We state the approximation ratio of the NRA algorithm in the following lemma.

Lemma 7. The approximation ratio of the NRA algorithm is $\pi_{NRA}^{\mathcal{U}} \geq \frac{Z-1}{Z-1+eZR^{1/(Z-1)}} R_4(S)$.

Proof. First, we define additional notations. Let $\mathcal{F}' \subseteq \mathcal{F}$ denote the set of unassigned flows by the end of Algorithm 5 and \mathcal{F}_v denote the set of unassigned flows right before considering VNF-node v by Algorithm 5. By slightly abusing the notation, we use OPT($\{v\}|\bar{\mathcal{F}}$) to denote the optimal resource allocation of VNF-node v considering only the subset of flows $\bar{\mathcal{F}}$. We have

$$\begin{aligned}
R_4(S) &\stackrel{(a)}{\leq} R_3(\mathcal{U}) \\
&\stackrel{(b)}{\leq} \sum_{v \in \mathcal{U}} \text{OPT}(\{v\}|\mathcal{F}) \\
&\stackrel{(c)}{=} \sum_{f \in \mathcal{F} \setminus \mathcal{F}'} \lambda_f + \sum_{v \in \mathcal{U}} \text{OPT}(\{v\}|\mathcal{F}') \\
&\stackrel{(d)}{=} \pi_{NRA}^{\mathcal{U}} + \sum_{v \in \mathcal{U}} \text{OPT}(\{v\}|\mathcal{F}') \\
&\stackrel{(e)}{\leq} \pi_{NRA}^{\mathcal{U}} + \sum_{v \in \mathcal{U}} \text{OPT}(\{v\}|\mathcal{F}_v) \\
&\stackrel{(f)}{\leq} \pi_{NRA}^{\mathcal{U}} + \sum_{v \in \mathcal{U}} \frac{eZ}{Z-1} R^{1/(Z-1)} \pi_{NRA}^{\{v\}} \\
&\leq \pi_{NRA}^{\mathcal{U}} + \frac{eZ}{Z-1} R^{1/(Z-1)} \pi_{NRA}^{\mathcal{U}} \\
&= \frac{Z-1+eZR^{1/(Z-1)}}{Z-1} \pi_{NRA}^{\mathcal{U}},
\end{aligned} \tag{22}$$

where (a) follows from Lemma 4; (b) holds because we consider each node individually with all flows \mathcal{F} ; (c) holds because we can consider what can be assigned from a subset of flows \mathcal{F}' and add to it all other flows $\mathcal{F} \setminus \mathcal{F}'$; (d) holds because flows $\mathcal{F} \setminus \mathcal{F}'$ are all assigned by the NRA algorithm; (e) holds because \mathcal{F}_v is a superset of \mathcal{F}' . For (f), the NRA 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 [15, Lemma 5.7, Theorem 5.1], so (f) holds. \square

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

5.4 Main Results

We state our main results in Theorems 3 and 4.

Theorem 3. *The SSG-PRA algorithm has an approximation ratio of $\frac{(e-1)(Z-1)}{4e^2 Z(kR)^{1/(Z-1)}}$ for Problem (P1) and becomes $\frac{e-1}{4e^2}$ when $Z \rightarrow \infty$.*

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:

$$\begin{aligned} R_4(S) &\stackrel{(a)}{\geq} \frac{1}{2}(1-1/e)OPT(Q4) \\ &\stackrel{(b)}{\geq} \frac{1}{4}(1-1/e)OPT(Q3) \\ &\stackrel{(c)}{=} \frac{1}{4}(1-1/e)OPT(Q1) \\ &\stackrel{(d)}{\geq} \frac{1}{4}(1-1/e)OPT(P1), \end{aligned} \quad (23)$$

where (a) is due to Theorem 2, (b) holds from Lemma 4, (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:

$$\begin{aligned} \pi_{PRA}^U &\stackrel{(a)}{\geq} \frac{Z-1}{eZ(kR)^{1/(Z-1)}} R_4(S) \\ &\stackrel{(b)}{\geq} \frac{(e-1)(Z-1)}{4e^2 Z(kR)^{1/(Z-1)}} OPT(P1), \end{aligned} \quad (24)$$

where (a) comes from the approximation ratio of the PRA algorithm in Lemma 6, and (b) holds from Eq. (23). Therefore, the result of Theorem 3 follows. \square

Theorem 4. *The SSG-NRA algorithm has an approximation ratio of $\frac{(e-1)(Z-1)}{4e(Z-1+eZR^{1/(Z-1)})}$ for Problem (P1) and becomes $\frac{e-1}{4e^2+4e}$ when $Z \rightarrow \infty$.*

Proof. The proof follows the same steps as the proof of Theorem 3. \square

6 NUMERICAL RESULTS

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 [16]. 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 we present the results of the optimal solution found by an ILP solver as a benchmark for comparisons only. While the ILP solver seems to work reasonably well for some problem instances considered in our simulations, the multi-VPRA problem is NP-hard in general (Lemma 1). That is, there is no guarantee that any problem instance can be efficiently solved, and it may take a prohibitively long time to solve the problem in the worst-case scenarios.

6.1 Evaluation Datasets

6.1.1 Abilene Dataset

We consider the Abilene dataset collected from an educational backbone network in North America [12]. 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 randomly chosen between 0 and 20 (i.e., $\delta_f^r \in [0, 20]$). The total available resource is set to the maximum total demand of flows d_{\max} multiplied by a scaling parameter $Z > 1$.

6.1.2 SNDlib Datasets

We also consider two other datasets from SNDlib [13]: 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.

6.2 Evaluation Results

We start with the Abilene dataset, where we study the effect of having different values of resource stretch Z and budget B . Remember 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 key 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 3 and 4 in that when the budget or Z is small, the SSG-NRA performs better and vice versa. Specifically, we start with Fig. 1(a)

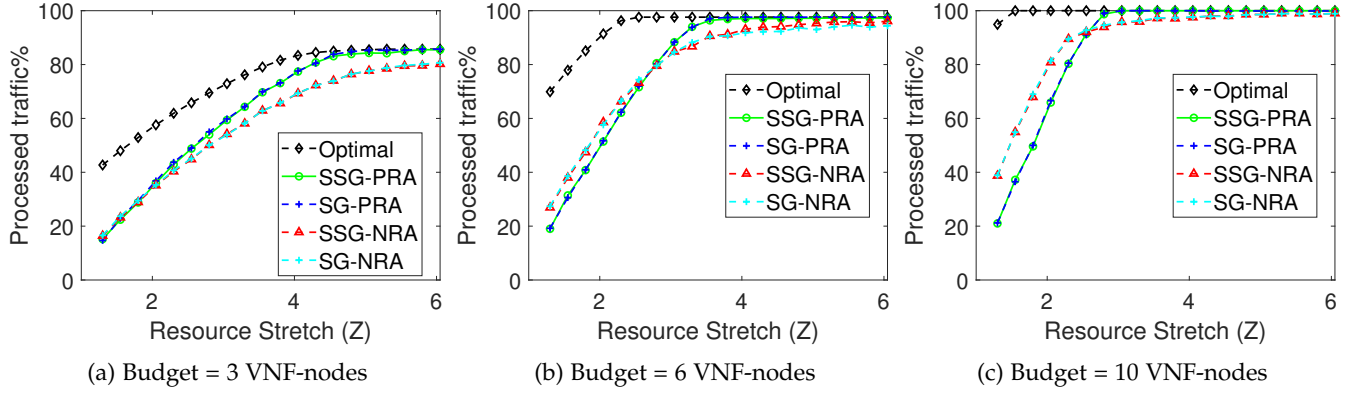


Fig. 1: Evaluation of Abilene dataset with different budget k and resource stretch Z

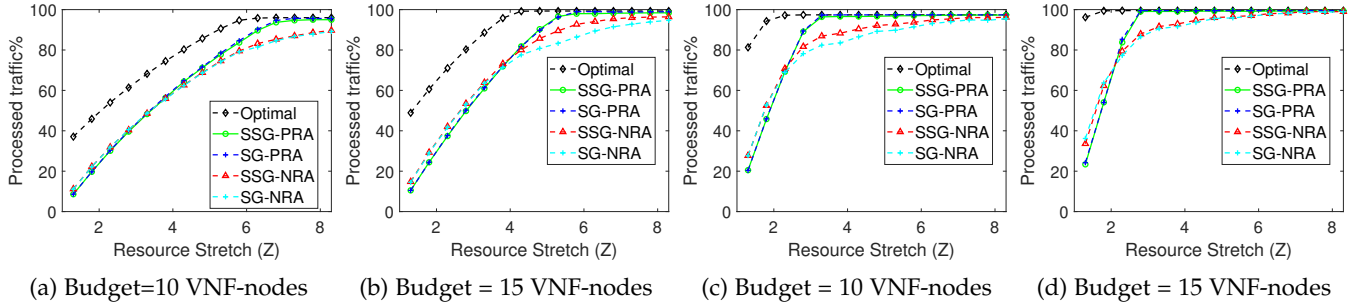


Fig. 2: Evaluation of Cost266 dataset (a-b) and ta2 dataset (c-d)

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 number of resources and nodes (i.e., $R|\mathcal{U}|$) is small anyway, it does not affect the performance of the SSG-PRA algorithm much. When Z becomes large (either by having larger amount of resources or by having flows with smaller demand to make $Z \geq 4$), the effect of the terms $R|\mathcal{U}|^{Z-1}$ and R^{Z-1} diminishes, but the effect of the constant term of the SSG-NRA algorithm remains, which corresponds to a slightly worse performance for larger Z . 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 the SSG-PRA algorithm when Z is small (i.e., $Z \leq 2.5$). This is because when Z is small and $R|\mathcal{U}|$ is large, there is a high chance that the stopping condition of the PRA algorithm is satisfied early although some nodes still have large 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 works 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 Figs. 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 work better in multiple occasions as for the SSG-NRA algorithm. That means even if our conjecture that $R_3(\mathcal{U})$ is submodular is correct, the loss by considering the second-level relaxation (i.e., the $1/2$ -approximation factor in Theorem 2) is negligible. However, the second-level relaxation is important as it allows to draw a connection to the sequence submodular theory and establish the performance guarantee of the SSG algorithm.

Fourth, The results suggest that in order to gain the best performance in term 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.

Furthermore, we extend the evaluation to other datasets with a larger number of nodes and flows in Fig. 2. We consider Cost266 dataset (37 nodes and 1332 flows) and ta2 dataset (65 nodes and 1869 flows). We consider two settings of budget of 10 and 15 VNF-nodes. Comparing with the proposed algorithms, we can see a similar trend to that of Fig. 1 in that the SSG-NRA algorithm works better for a smaller Z and vice-versa for the SSG-PRA algorithm. Comparing both algorithms with the optimal solution, the proposed algorithms are also within $1/2$ of the value achieved by the optimal solution. In addition, we can see that the heuristics and the proposed algorithms perform very similarly to each

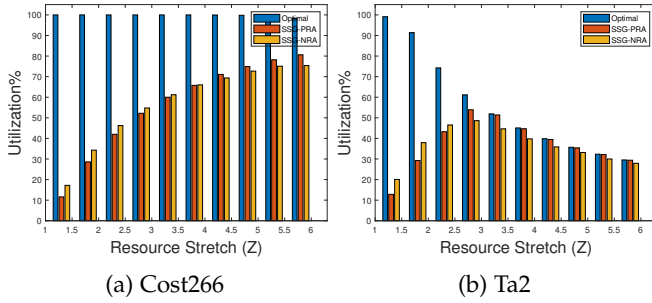


Fig. 3: VNF-node utilization with budget = 10 VNF-nodes

other and that no algorithm constantly dominates the other. We note that although the resource stretch Z is the same for Cost266 dataset and ta2 dataset, the actual amount of resources is different because the maximum flow rate of ta2 dataset is 140 times more than that of the cost255 dataset. However, the total flow rates of ta2 dataset are 50 times less than the total flow rates of Cost266 dataset. That explains why for a similar budget, we have a better performance for all algorithms under ta2 dataset (e.g., Fig. 2(c)) compared to Cost266 dataset (e.g., Fig. 2(a)).

In addition, we also study the impact on the resource utilization of VNF-nodes. For each VNF-node, we compute the utilization per resource type as the amount of used resource divided by the total amount of each available resource, which is then averaged over all resource types and over all VNF-nodes. We compare our proposed algorithms with the optimal solution and present these results in Fig. 3. The SG-PRA and SG-NRA algorithms exhibit similar results to our algorithms as the utilization primarily depends on the resource allocation component, which is the same for both the SG-PRA/SG-NRA algorithms and our proposed algorithms. We can observe that the utilization increases as the resource stretch increases, mainly due to the increase in the total processed traffic as shown in Figs. 2(a) and 2(c). However, the utilization naturally decreases once close to 100% of the traffic has been processed (see Fig. 3(b)). This is because more resource becomes available but not used.

Finally, we present the average running time of all algorithms in Table 2. We observe that all algorithms have a small running time; even for the ILP, solved using the Gurobi solver, the running time is small as well. As mentioned earlier, while the ILP solver seems to work reasonably well for the problem instances considered in our simulations, there is no guarantee that the ILP can be efficiently solved due to its NP-hardness. Therefore, we use it as a benchmark for comparison purposes only.

7 RELATED WORK

The transition to NFV often happens in two phases: the planning phase and the production phase. In the planning phase, the main focus is on deciding where to introduce the NFV capabilities to maximize the benefit (i.e., the placement problem). Since this phase happens before the production phase, we can assume certain behaviors about the production phase and use some historical traces to project flow demands and routing. In the production phase, finer-grained optimization will be considered, such as flow admission and

routing. In this work, we focus on the planning phase and assume a fixed routing. In the following, we will discuss related work on several relevant topics.

Placement: The placement problem has been considered in different domains such as NFV (e.g., [7]), software-defined networks (SDN) (e.g., [5]), and edge cloud computing (e.g., [17]). In NFV, several studies (e.g., [6], [18], [19]) consider the placement of a minimum number of VNF instances to cover all flows. A single type of network functions is considered in [6], [18], [20], [21], and the case of multiple network functions is considered in [19], [22]–[27]. However, these work neglects either the budget constraint or the multi-dimensional resource allocation. The work in [28] considers the placement of middleboxes to make the shortest path between communicating pairs under a threshold. A generalized version of [28] is presented in [29], which considers the problem of joint service placement and request routing in mobile edge computing networks. Again, this work does not consider multiple network functions or budget constraint. Other work considers different objectives, such as delay minimization (e.g. [30], [31]), energy efficiency (e.g., [32], [33]), fault tolerance (e.g., [34]), and revenue maximization (e.g., [35], [36]).

Budget-constrained Resource Allocation: Budget-constrained resource allocation has been considered in prior studies (e.g., [5], [7], [17], [37], [38]). In [37], the authors consider the problem of request routing and capacity allocation with the objective of maximizing the total traffic of admitted requests under a limited budget. They assume that all nodes in the network already have NFV capabilities and focus on maximizing the utilization of the budget in the case that admitting a request has a certain cost. In the SDN domain, the work in [5] considers the placement of SDN-enabled routers to maximize the total processed traffic. They consider a budget constraint but neglect the resource constraint. Similarly, in the work on edge cloud computing [17], [38], 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. In our previous work [7], we study the NFV placement problem where budget, resource, and fully processed flow constraints are considered, but we only consider one type of network function and one type of resource.

Multi-dimensional Resource Allocation: To the best of our knowledge, the multi-dimensional setting has rarely been considered except in a limited number of studies. In [39], 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 [40] focuses only on request admission and routing. The work of [41] 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. In [42], the authors consider servers with two types of resources, and their objective is to serve all network flows by placing network functions on these servers with minimum cost. Our work considers all three constraints of budget, resource, and fully processed flows, as well as the multi-dimensional setting.

Dataset \ Algorithm	ILP	SSG-PRA	SSG-NRA	SG-PRA	SG-NRA
cost266	17	8	8	9	9
ta2	10	12	11	12	12

TABLE 2: Running time (seconds)

Sequence Submodularity: The concept of sequence (or string) submodularity is a generalization of submodularity, which has been recently introduced in several studies (e.g., [9], [10], [14], [43]). It models objective functions that depend on the sequence of actions. It has been shown in [14] that a simple greedy algorithm can achieve an approximation ratio of $(1 - 1/e)$ for maximizing forward-monotone, backward-monotone, and sequence-submodular functions. To the best of our knowledge, we are the first to utilize the concept of sequence submodularity for the placement problem in NFV. Although in [10], the backward-monotone property is not assumed, per our investigation, there is a crucial step in the proof that requires this property. However, the objective function of our placement subproblem (i.e., Problem (Q4) defined in Section 4.2) does not satisfy this backward-monotone property, rendering the results of [10] inapplicable to our problem. To address this new challenge, we introduce the concept of approximate backward-monotone and extend the previous result to this generalization.

Preliminary Version: A preliminary version of this work was presented at IEEE ICNP 2019 [1]. This extended journal version includes a correction of Lemma 4 in [1] (Theorem 2 in this version). In [1], we applied the results in [10, Theorem 3], which states that for an objective function that is forward-monotone and sequence-submodular, the greedy algorithm (Algorithm 3) achieves an approximation ratio of $(1 - 1/e)$. However, it turns out that we also need the objective function to satisfy a backward-monotone property (defined in Section 5.1) in order to have the $(1 - 1/e)$ approximation. This is consistent with the results in other highly relevant work (e.g., [14]). We show that our objective function does not satisfy the backward-monotone property but satisfies an approximate version of the property. Based on that, we derive the approximation ratio of the greedy algorithm for forward-monotone, approximate backward-monotone, and sequence-submodular functions and apply the result to our problem.

8 CONCLUSION AND FUTURE WORK

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 handle 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 becomes close to the optimal solution earlier.

In our future work, we would like to consider the following interesting directions. First, we will consider service function chaining, where the network functions required for each flow must be in a specific order. Second, we will also consider heterogeneous node cost. It is worth noting that in our previous work [7], we have considered heterogeneous node cost where a single type of resource and a single type of network function are assumed, and we were able to develop efficient algorithms with approximation ratio guarantees. However, it is unclear whether one can extend such algorithms to the multi-dimensional setting and still achieve certain performance guarantees. Finally, it would be interesting to explore the possibility of developing polynomial-time approximation schemes (PTAS) for the resource allocation subproblem and the overall problem we consider.

9 PROOFS

9.1 Proof of Lemma 2

Proof. Based on Algorithm 1, nodes in S_3 will be considered first when solving for sequence S_1 and S_2 , and in the same order. Therefore, the amount of assigned traffic to nodes in S_3 will be the same. \square

9.2 Proof of Lemma 3

Proof. First, we can express the right-hand side of Eq. (14) as follows:

$$\begin{aligned} & R_4(S_1 \oplus u | \lambda) - R_4(S_1 | \lambda) \\ &= \sum_{i=1}^{|S_1|} \hat{y}_i(S_1 \oplus u) + \hat{y}_{|S_1 \oplus u|}(S_1 \oplus u) - \sum_{i=1}^{|S_1|} \hat{y}_i(S_1) \\ &= \hat{y}_{|S_1 \oplus u|}(S_1 \oplus u), \end{aligned}$$

where the last equality holds because $\hat{y}_i(S_1 \oplus u) = \hat{y}_i(S_1)$ for $i \leq |S_1|$ from Lemma 2. Recall from Algorithm 1 that $\hat{y}_{|S_1 \oplus u|}(S_1 \oplus u)$ is the value of the optimal solution of Problem (6) for node u given $[\hat{y}_1(S_1 \oplus u), \dots, \hat{y}_{|S_1|}(S_1 \oplus u), 0]$, which we include in the following to easily navigate the proof:

$$\begin{aligned} & \text{maximize}_{\mathbf{X}(S_1 \oplus u)} \sum_{f \in \mathcal{F}} x_f^u \\ & \text{subject to} \\ & \sum_{v \in \mathcal{V}_f \cap \mathcal{V}(S_1 \oplus u)} x_f^v \leq \lambda_f, \quad \forall f \in \mathcal{F} \text{ and } v \in \mathcal{V}_f, \quad (25a) \\ & x_f^v = 0, \quad \forall f \in \mathcal{F} \text{ and } v \notin \mathcal{V}_f, \quad (25b) \end{aligned}$$

$$\sum_{\phi \in \Phi} \beta_{\phi}^r \sum_{f \in \mathcal{F}(\phi)} x_f^{v_i} \leq c_{v_i}^r, \quad \forall r \in \mathcal{R}, i = 1, \dots, |S_1 \oplus u|, \quad (25c)$$

$$\sum_{f \in \mathcal{F}} x_f^{v_i} = \hat{y}_i(S_1 \oplus u), \quad i = 1, \dots, |S_1|. \quad (25d)$$

On the other hand, $R_4(u | (\lambda - \hat{x}(S_2)))$ corresponds to solving the following problem:

$$\begin{aligned} & \underset{\mathbf{X}(u)}{\text{maximize}} \quad \sum_{f \in \mathcal{F}} x_f^u \\ & \text{subject to} \\ & x_f^u \leq \lambda_f - \hat{x}_f(S_2), \quad \forall f \in \mathcal{F} \text{ and } u \in \mathcal{V}_f \\ & x_f^u = 0, \quad \forall f \in \mathcal{F} \text{ and } u \notin \mathcal{V}_f, \\ & \sum_{\phi \in \Phi} \beta_{\phi}^r \sum_{f \in \mathcal{F}(\phi)} x_f^u \leq c_u^r, \quad \forall r \in \mathcal{R}. \end{aligned} \quad (26)$$

Recall that $\hat{\mathbf{X}}(S_2)$ denotes the flow assignment matrix of sequence S_2 at the end of Algorithm 1; we also use $\hat{\mathbf{X}}(u)$ to denote the flow assignment matrix of sequence (u) after solving Problem (26). We construct a flow assignment matrix $\bar{\mathbf{X}}(S_1 \oplus u)$ by concatenating the flow assignment of the first $|S_1|$ nodes in $\hat{\mathbf{X}}(S_2)$ with the flow assignment of node u in $\hat{\mathbf{X}}(u)$, i.e., we let $\bar{\mathbf{X}}(S_1 \oplus u) = [\hat{\mathbf{X}}(S_2)_{(:,1:|S_1|)}, \hat{\mathbf{X}}(u)]$. It is easy to see that the value of the constructed flow assignment matrix $\bar{\mathbf{X}}(S_1 \oplus u)$ is equal to $R_4(u | (\lambda - \hat{x}(S_2)))$. To conclude the lemma, we need to show that $\bar{\mathbf{X}}(S_1 \oplus u)$ is a feasible solution to Problem (25). We do so by showing that each constraint of Problem (25) is satisfied as follows.

1) Constraint (25a). For each flow f , we have

$$\begin{aligned} & \sum_{v \in \mathcal{V}_f \cap \mathcal{V}(S_1 \oplus u)} \bar{x}_f^v(S_1 \oplus u) \\ & \stackrel{(a)}{=} \sum_{i=1}^{|S_1|} \bar{x}_f^{v_i}(S_1 \oplus u) + \bar{x}_f^u(S_1 \oplus u) \\ & \stackrel{(b)}{=} \sum_{i=1}^{|S_1|} \hat{x}_f^{v_i}(S_2) + \hat{x}_f^u((u)) \\ & \stackrel{(c)}{\leq} \sum_{i=1}^{|S_1|} \hat{x}_f^{v_i}(S_2) + \lambda_f - \hat{x}_f(S_2) \\ & \stackrel{(d)}{=} \sum_{i=1}^{|S_1|} \hat{x}_f^{v_i}(S_2) + \lambda_f - \sum_{i=1}^{|S_2|} \hat{x}_f^{v_i}(S_2) \\ & \stackrel{(e)}{\leq} \lambda_f, \end{aligned}$$

where (a) holds because $\bar{x}_f^v(S_1 \oplus u) = 0$ for $v \notin \mathcal{V}_f$; (b) follows from the way we constructed $\bar{\mathbf{X}}(S_1 \oplus u)$; (c) holds because $\hat{x}_f^u(u) \leq \lambda_f - \hat{x}_f(S_2)$ from Problem (26); (d) holds from the definition of $\hat{x}_f(S_2)$ in Eq. (10); (e) holds because $|S_1| \leq |S_2|$.

2) Constraint (25b). This constraint is satisfied by the feasibility of $\hat{\mathbf{X}}(S_2)$ and $\hat{\mathbf{X}}(u)$.

3) Constraint (25c). The flow assignment matrix $\hat{\mathbf{X}}(S_2)_{(:,1:|S_1|)}$ satisfies Constraint (25c) for sequence S_1 and does not assign any traffic to node u . Similarly, the flow assignment matrix $\hat{\mathbf{X}}(u)$ satisfies Constraint (25c) for node u without affecting the assigned traffic to sequence S_1 . Therefore, the constructed flow assignment matrix $\bar{\mathbf{X}}(S_1 \oplus u)$ satisfies Constraint (25c) for sequence $S_1 \oplus u$.

4) Constraint (25d). For $i = 1, \dots, |S_1|$, we have

$$\begin{aligned} \sum_{f \in \mathcal{F}} \bar{x}_f^{v_i}(S_1 \oplus u) & \stackrel{(a)}{=} \sum_{f \in \mathcal{F}} \hat{x}_f^{v_i}(S_2) \\ & \stackrel{(b)}{=} \hat{y}_i(S_2) \\ & \stackrel{(c)}{=} \hat{y}_i(S_1 \oplus u), \end{aligned}$$

where (a) follows from the way we constructed $\bar{\mathbf{X}}(S_1 \oplus u)$; (b) holds from the feasibility of $\hat{\mathbf{X}}(S_2)_{(:,1:|S_1|)}$; (c) holds because $\hat{y}_i(S_2) = \hat{y}_i(S_1 \oplus u)$ for $i \leq |S_1|$ from Lemma 2. \square

9.3 Proof of Lemma 4

Proof. Define $\hat{\mathbf{x}}^{\min}(\mathcal{U}) \triangleq \min(\hat{\mathbf{x}}(\mathcal{U}), \hat{\mathbf{x}}(S))$ and $\hat{\mathbf{x}}^{\max}(\mathcal{U}) \triangleq \max(\hat{\mathbf{x}}(\mathcal{U}) - \hat{\mathbf{x}}(S), 0)$. Note that $\hat{\mathbf{x}}^{\min}(\mathcal{U}) + \hat{\mathbf{x}}^{\max}(\mathcal{U}) = \hat{\mathbf{x}}(\mathcal{U})$. To show that $\frac{1}{2}R_3(\mathcal{U}) \leq R_4(S)$, we prove the following:

(A) $\sum_{j=1}^F \hat{x}_j^{\min}(\mathcal{U}) \leq R_4(S)$;

(B) $\sum_{j=1}^F \hat{x}_j^{\max}(\mathcal{U}) \leq R_4(S)$.

By combining (A) and (B), we get that

$$\begin{aligned} R_3(\mathcal{U}) &= \sum_{j=1}^F \hat{x}_j(\mathcal{U}) \\ &= \sum_{j=1}^F \hat{x}_j^{\min}(\mathcal{U}) + \sum_{j=1}^F \hat{x}_j^{\max}(\mathcal{U}) \\ &\leq 2R_4(S). \end{aligned}$$

To show that (A) holds, we have

$$\sum_{j=1}^F \hat{x}_j^{\min}(\mathcal{U}) \stackrel{(a)}{\leq} \sum_{j=1}^F \hat{x}_j(S) \stackrel{(b)}{=} R_4(S), \quad (27)$$

where (a) holds since $\hat{\mathbf{x}}^{\min}(\mathcal{U}) \leq \hat{\mathbf{x}}(S)$; (b) holds from the definition of function R_4 in Eq. (11).

Next, we show that (B) holds. Recall that by definition, the flow rate vector $\hat{\mathbf{x}}(\mathcal{U})$ can be fully assigned to nodes \mathcal{U} . Therefore, any vector $\bar{\mathbf{x}}(\mathcal{U}) \leq \hat{\mathbf{x}}(\mathcal{U})$ can be fully assigned to nodes \mathcal{U} as well, and we can establish the following:

$$R_3(\mathcal{U} | \bar{\mathbf{x}}(\mathcal{U})) = \sum_{j=1}^F \bar{x}_j(\mathcal{U}). \quad (28)$$

Moreover, we have

$$\begin{aligned} \sum_{j=1}^F \hat{x}_j^{\max}(\mathcal{U}) & \stackrel{(a)}{=} R_3(\mathcal{U} | \hat{\mathbf{x}}^{\max}(\mathcal{U})) \\ & \stackrel{(b)}{\leq} R_3(\mathcal{U} | (\lambda - \hat{\mathbf{x}}(S))) \\ & \stackrel{(c)}{\leq} \sum_{v \in \mathcal{U}} R_3(\{v\} | (\lambda - \hat{\mathbf{x}}(S))) \\ & \stackrel{(d)}{=} \sum_{i=1}^{|S|} R_4(v_i | (\lambda - \hat{\mathbf{x}}(S))) \\ & \stackrel{(e)}{\leq} \sum_{i=1}^{|S|} (R_4((v_1, \dots, v_{i-1}, v_i) | \lambda) \\ & \quad - R_4((v_1, \dots, v_{i-1}) | \lambda)) \\ & = R_4(S), \end{aligned} \quad (29)$$

where (a)-(e) hold for the following. (a) follows from Eq. (28). For (b), recall that $\hat{x}^{\max}(\mathcal{U}) = \max(\hat{x}(\mathcal{U}) - \hat{x}(S), 0)$. Since $\hat{x}(\mathcal{U}) \leq \lambda$, we get that $\hat{x}^{\max}(\mathcal{U}) \leq \lambda - \hat{x}(S)$. By applying Eq. (12), we get that (b) holds. (c) holds because we consider each node in \mathcal{U} individually with the same traffic rate vector $(\lambda - \hat{x}(S))$, so the resource allocation in the solution to nodes \mathcal{U} is a feasible solution to each individual node. For (d), note that when applying Algorithm 1 for a singleton sequence, the equality constraints of Problem (6) are irrelevant and function $R_4(\{(v_i)\})$ and $R_3(\{v_i\})$ will get the same result, so (d) holds. (e) holds from Lemma 3, where $(v_1, \dots, v_{i-1}) \preceq S$ and node $v_i \notin \mathcal{V}((v_1, \dots, v_{i-1}))$.

Finally, we show that $R_4(S) \leq R_3(\mathcal{U})$. We can see that the flow assignment matrix $\hat{X}(S)$ satisfies the flow rate constraint of all flows and the resources constraint of nodes in S . Since $S \in \mathcal{P}(\mathcal{U})$, then $\hat{X}(S) \in \mathcal{X}(\mathcal{U})$, and we get that $\hat{X}(S)$ is a feasible solution to Problem (Q2). Therefore, the inequality holds. \square

9.4 Proof of Theorem 1

Proof. First, we present Lemma 8, which shows that adding a node according to Line 2 of Algorithm 3 yields a marginal value that is greater than or equal to the average of adding any other sequence of nodes. Let $S^i = (v_1, \dots, v_i)$ denote the sequence of nodes selected by Algorithm 3 in the first i iterations, with S^0 to denote an empty sequence. Also, for any two sequences S_1 and S_2 , we define $h(S_2|S_1) \triangleq h(S_1 \oplus S_2) - h(S_1)$.

Lemma 8 ([10]). *Let v_i denote the node selected in iteration i by Algorithm 3. For any sequence $S' \in \mathcal{H}$, we have $h(v_i|S^{i-1}) \geq \frac{1}{|S'|} h(S'|S^{i-1})$.*

Next, let S^* denote the optimal sequence. Since function h is forward-monotone, we can assume that the length of sequence S^* is exactly k . Using Lemma 8, we have

$$\begin{aligned} h(v_i|S^{i-1}) &\geq \frac{1}{k} h(S^*|S^{i-1}) \\ &= \frac{1}{k} (h(S^{i-1} \oplus S^*) - h(S^{i-1})) \\ &\geq \frac{1}{k} (\alpha h(S^*) - h(S^{i-1})), \end{aligned} \quad (30)$$

where the last inequality holds because function h is α -backward-monotone. By rewriting Eq. (30), we have

$$h(S^i) - h(S^{i-1}) \geq \frac{1}{k} (\alpha h(S^*) - h(S^{i-1})), \quad (31)$$

which is equivalent to

$$h(S^i) \geq \frac{\alpha}{k} h(S^*) + (1 - \frac{1}{k}) h(S^{i-1}), \quad (32)$$

Writing Eq. (32) for $i = k$ and expanding it yields

$$\begin{aligned} h(S^k) &\geq \frac{\alpha}{k} h(S^*) \\ &\quad + \frac{\alpha}{k} (1 - \frac{1}{k}) h(S^*) \\ &\quad + \frac{\alpha}{k} (1 - \frac{1}{k})^2 h(S^*) \\ &\quad + \dots \\ &\quad + \frac{\alpha}{k} (1 - \frac{1}{k})^{k-1} h(S^*) + (1 - \frac{1}{k})^k h(S^0) \\ &= \alpha (1 - (1 - \frac{1}{k})^k) h(S^*) \\ &\geq \alpha (1 - \frac{1}{e}) h(S^*), \end{aligned} \quad (33)$$

where the last inequality holds because $(1 - \frac{1}{k})^k \leq \frac{1}{e}$. \square

9.5 Proof of Lemma 5

Proof. First, for any two sequences S_1, S_2 , we assume that $S_1 \oplus S_2$ has no repeated nodes, because if otherwise, we can remove the later appearance of the same node without affecting the value of $R_4(S_1 \oplus S_2)$.

Now, we proceed with the proof. First, we show that function $R_4(S)$ is forward-monotone (i.e., satisfies Eq. (16)). Since $S_1 \preceq S_1 \oplus S_2$, then according to Lemma 2, the assigned traffic to nodes in S_1 will be the same for the two sequences S_1 and $S_1 \oplus S_2$. Adding additional nodes to sequence S_1 will not affect the amount of traffic already assigned to nodes in S_1 , and the minimum that can be assigned to any node is zero. So, Eq. (16) is satisfied.

Next, we show that function $R_4(S)$ is $\frac{1}{2}$ -backward-monotone (i.e., satisfies Eq. (18) with $\alpha = \frac{1}{2}$). We have the following:

$$\begin{aligned} R_4(S_1 \oplus S_2) &\stackrel{(a)}{\geq} \frac{1}{2} R_3(\mathcal{V}(S_1 \oplus S_2)) \\ &\stackrel{(b)}{\geq} \frac{1}{2} R_3(\mathcal{V}(S_2)) \\ &\stackrel{(c)}{\geq} \frac{1}{2} R_4(S_2) \end{aligned} \quad (34)$$

where (a) follows from Lemma 4. For (b), note that function R_3 is monotonically nondecreasing because adding an additional VNF-node does not reduce the amount of flows that can be processed. Since $\mathcal{V}(S_2)$ is a subset of $\mathcal{V}(S_1 \oplus S_2)$ and function R_3 is nondecreasing, then (b) holds. (c) holds from Lemma 4.

Finally, we show that function $R_4(S)$ is sequence-submodular (i.e., satisfies Eq. (15)). For Eq. (15) to be satisfied, we need to show that $R_4(S_1 \oplus u) - R_4(S_1) \geq R_4(S_2 \oplus u) - R_4(S_2)$ for any $S_1 \preceq S_2$ and $u \in \mathcal{V}$. We distinguish between two cases:

Case I: node $u \in \mathcal{V}(S_2)$. In this case, according to the definition of function R_4 , a repeated node has zero marginal gain (i.e., $R_4(S_2 \oplus u) - R_4(S_2) = 0$). Since function R_4 is forward-monotone, then we also get that $R_4(S_1 \oplus u) - R_4(S_1) \geq 0$. Therefore, function R_4 is sequence-submodular.

Case II: node $u \notin \mathcal{V}(S_2)$, which also implies that $u \notin \mathcal{V}(S_1)$. Let $\hat{X}(S_2 \oplus u)$ denote the flow assignment matrix of sequence $S_2 \oplus u$ at the end of Algorithm 1. Let $\hat{x}(S_2 \oplus u)$

be the flow rate vector extracted from $\hat{\mathbf{X}}(S_2 \oplus u)$. We define $\bar{\mathbf{X}}(S_2) \triangleq [\hat{\mathbf{X}}(S_2 \oplus u)_{(:,1:|S_2|)}]$. Note that $\bar{\mathbf{X}}(S_2)$ is a possible realization of the flow assignment matrix of sequence S_2 at the end of Algorithm 1. Let $\bar{\mathbf{x}}(S_2)$ be the flow rate vector extracted from $\bar{\mathbf{X}}(S_2)$. The proof proceeds as follows:

$$\begin{aligned}
 & R_4(S_2 \oplus u) - R_4(S_2) \\
 &= \sum_{i=1}^{|S_2|} \hat{y}_i(S_2 \oplus u) + \hat{y}_{|S_2 \oplus u|}(S_2 \oplus u) - \sum_{i=1}^{|S_2|} \hat{y}_i(S_2) \\
 &\stackrel{(a)}{=} \hat{y}_{|S_2 \oplus u|}(S_2 \oplus u) \\
 &= \sum_{j=1}^F \hat{x}_j(S_2 \oplus u) - \bar{x}_j(S_2) \\
 &\stackrel{(b)}{=} R_4(u)(\hat{\mathbf{x}}(S_2 \oplus u) - \bar{\mathbf{x}}(S_2)) \\
 &\stackrel{(c)}{\leq} R_4(u)(\boldsymbol{\lambda} - \bar{\mathbf{x}}(S_2)) \\
 &\stackrel{(d)}{\leq} R_4(S_1 \oplus u) - R_4(S_1),
 \end{aligned} \tag{35}$$

where (a) holds because $\hat{y}_i(S_2 \oplus u) = \hat{y}_i(S_2)$ for $i \leq |S_2|$ from Lemma 2; for (b), the flow rate vector $(\hat{\mathbf{x}}(S_2 \oplus u) - \bar{\mathbf{x}}(S_2))$ is what has been assigned to node u while considering sequence $S_2 \oplus u$, so it can also be assigned to node u when considering the singleton sequence consisting of only node u ; (c) holds from Eq. (13) since $\hat{\mathbf{x}}(S_2 \oplus u) \leq \boldsymbol{\lambda}$; (d) holds from Lemma 3 since $S_1 \preceq S_2$ and node $u \notin \mathcal{V}(S_1)$. \square

9.6 Function R_4 $\frac{1}{2}$ -backward-monotone: Tight Bound Proof

We show that the lower bound of $\frac{1}{2}$ in Eq. (34) is tight through the following problem instance. Consider three nodes v_1, v_2, v_3 , three flows f_1, f_2, f_3 , and two types of resources r_1, r_2 . Assume the following: the traffic rate of each flow is $\epsilon_1, 1$, and $1 + \epsilon_2$, respectively, with $\epsilon_1 > \epsilon_2$ for arbitrary small ϵ_1 and ϵ_2 ; the amount of each resource at each node is c ; the path of each flow is (v_1, v_2) , (v_2) , and (v_2, v_3) , respectively. Let $\delta_f^r \triangleq \sum_{\phi \in \Phi_f} \beta_\phi^r$ be the total amount of resource r needed to process a unit of flow f by the set of network functions Φ_f . Assume that the following holds:

- i) $\delta_{f_1}^{r_1} \times \lambda_{f_1} + \delta_{f_2}^{r_1} \times \lambda_{f_2} = c$,
- ii) $\delta_{f_1}^{r_2} \times \lambda_{f_1} + \delta_{f_2}^{r_2} \times \lambda_{f_2} = c$,
- iii) $\delta_{f_3}^{r_1} \times \lambda_{f_3} = \delta_{f_3}^{r_2} \times \lambda_{f_3} = c$,
- iv) $\delta_{f_2}^{r_1} > \delta_{f_3}^{r_1}$.

It can be verified that if the above assumptions hold, then $\mathbf{y}((v_1, v_2, v_3)) = [\epsilon_1, 1 + \epsilon_2, 0]$, while $\mathbf{y}((v_2, v_3)) = [1 + \epsilon_1, 1 + \epsilon_2]$. As a result, we get that $R_4(v_1, v_2, v_3) = 1 + \epsilon_1 + \epsilon_2$ and $R_4(v_2, v_3) = 2 + \epsilon_1 + \epsilon_2$, for arbitrary small ϵ_1 and ϵ_2 .

The following is an example (also presented in Fig. 4) that satisfies the aforementioned assumptions. Consider the traffic rate of each flow to be 0.02, 1, and 1.01, respectively, and the amount of each resource at each node to be 10. Also, assume that the demand of each resource by each flow (i.e., $[\delta_f^{r_1}, \delta_f^{r_2}]$) to be $[40, 100]$, $[9.2, 8]$, and $[9.9009, 9.9009]$, respectively. First, we evaluate $R_4(v_2, v_3)$. We start with an initial resource allocation $\mathbf{y}((v_2, v_3)) = [0, 0]$. We evaluate $R_4(v_2, v_3)$ using Algorithm 1. We start with node v_2 and solve Problem (6) for node v_2 given $\mathbf{y}((v_2, v_3))$. The algorithm will assign flow f_1 and f_2 to node v_2 , and the total

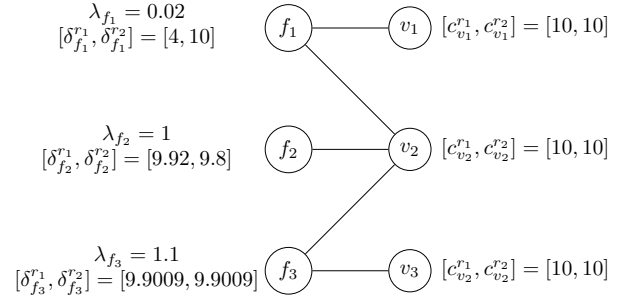


Fig. 4: An example to show that the bound of $\frac{1}{2}$ in Eq. (34) is tight. Computing $R_4(v_2, v_3)$ results in the following assignment vector $[1.02, 1.01]$, while computing $R_4(v_1, v_2, v_3)$ results in the following assignment vector $[0.02, 1.01, 0]$.

traffic assigned to node v_2 is equal to 1.02. If we try to assign any combinations of flows f_1 and f_3 or of flows f_2 and f_3 , the most we can assign to node v_2 is 1.01. Therefore, we update $\mathbf{y}((v_2, v_3)) = [1.02, 0]$. Next, we solve Problem (6) for node v_3 given $\mathbf{y}((v_2, v_3))$. The flows assigned to node v_2 will remain the same and flow f_3 will be assigned to node v_3 . The result is $\mathbf{y}((v_2, v_3)) = [1.02, 1.01]$. Therefore, the value of $R_4(v_2, v_3)$ is 2.03.

Next, we evaluate $R_4(v_1, v_2, v_3)$. The initial resource allocation is $\mathbf{y}((v_1, v_2, v_3)) = [0, 0, 0]$. We solve Problem (6) for node v_1 given $\mathbf{y}((v_1, v_2, v_3))$. The result is that flow f_1 will be assigned to node v_1 and $\mathbf{y}((v_1, v_2, v_3))$ becomes $[0.02, 0, 0]$. We repeat the same steps for node v_2 . The maximum traffic that can be assigned to node v_2 given $\mathbf{y}((v_1, v_2, v_3))$ is 1.01, and the only way to achieve that is by assigning flow f_3 to node v_2 . If we try to assign portions of flows f_2 and f_3 , we will always end up with total less than 1.01. The reason for this is that $\delta_{f_3}^{r_1} \times \lambda_{f_3} = 10$ and if we want to replace a unit of flow f_3 with a unit of flow f_2 , we will not be able to do so because flow f_2 is more expensive than flow f_3 (i.e., $\delta_{f_2}^{r_1} > \delta_{f_3}^{r_1}$). We update $\mathbf{y}((v_1, v_2, v_3))$ to become $[0.02, 1.01, 0]$. Finally, we solve Problem (6) for node v_3 given $\mathbf{y}((v_1, v_2, v_3))$. In order to satisfy the equality constraints of nodes v_1 and v_2 , we have to assign flow f_1 to node v_1 and flow f_3 to node v_2 , as explained before. Therefore, we will not be able to assign any traffic to node v_3 . In this case, the value of $R_4((v_1, v_2, v_3))$ is 1.03.

REFERENCES

- [1] G. Sallam, Z. Zheng, and B. Ji, "Placement and allocation of virtual network functions: Multi-dimensional case," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–11.
- [2] M. Latva-aho, K. Leppänen, F. Clazzer, and A. Munari, "Key drivers and research challenges for 6g ubiquitous wireless intelligence," 2020.
- [3] 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.
- [4] Amdocs, "Bringing NFV to Life - Technological and Operational Challenges in Implementing NFV," *White paper*, 2016.
- [5] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassioulas, "One step at a time: Optimizing sdn upgrades in isp networks," in *Proceedings of IEEE INFOCOM*, 2017.

- [6] 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.
- [7] G. Sallam and B. Ji, "Joint placement and allocation of vnf nodes with budget and capacity constraints," *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1238–1251, 2021.
- [8] 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.
- [9] 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.
- [10] S. Alaei, A. Makhdomi, and A. Malekian, "Maximizing sequence-submodular functions and its application to online advertising," *arXiv preprint arXiv:1009.4153*, 2010.
- [11] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.
- [12] "Abilene dataset, <http://www.cs.utexas.edu/~yzhang/research/abilenetm>," 2010.
- [13] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.
- [14] M. Streeter and D. Golovin, "An online algorithm for maximizing submodular functions," in *Advances in Neural Information Processing Systems*, 2009, pp. 1577–1584.
- [15] 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.
- [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] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] K. Poularakis, J. Llorca, A. M. Tulino, and L. Tassiulas, "Approximation algorithms for data-intensive service chain embedding," in *Proceedings of MobiHoc 2020*, 2020, pp. 131–140.
- [27] L. Gu, J. Hu, D. Zeng, S. Guo, and H. Jin, "Service function chain deployment and network flow scheduling in geo-distributed data centers," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2587–2597, 2020.
- [28] T. Lukovszki, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *Journal of Parallel and Distributed Computing*, 2018.
- [29] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, 2020.
- [30] G. Sun, R. Zhou, J. Sun, H. Yu, and A. V. Vasilakos, "Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization," *IEEE Internet of Things Journal*, 2020.
- [31] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu, "Efficient algorithms for delay-aware nfv-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2050–2066, 2020.
- [32] C. Zhang, X. Wang, A. Dong, Y. Zhao, Q. He, and M. Huang, "Energy efficient network service deployment across multiple sdn domains," *Computer Communications*, 2020.
- [33] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for vnf placement and chaining," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017, pp. 579–588.
- [34] G. Yuan, Z. Xu, B. Yang, W. Liang, W. K. Chai, D. Tuncer, A. Galis, G. Pavlou, and G. Wu, "Fault tolerant placement of stateful vnfs and dynamic fault recovery in cloud networks," *Computer Networks*, vol. 166, p. 106953, 2020.
- [35] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vspace: Vnf simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, 2018.
- [36] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, "Dynamic vnf placement, resource allocation and traffic routing in 5g," *Computer Networks*, vol. 188, p. 107830, 2021.
- [37] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, "Qos-aware vnf placement and service chaining for iot applications in multi-tier mobile edge networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–27, 2020.
- [38] V. Farhadi, F. Mehmeti, T. He, T. F. La Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 779–792, 2021.
- [39] 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.
- [40] 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.
- [41] 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.
- [42] Z. Luo and C. Wu, "An online algorithm for vnf service chain scaling in datacenters," *IEEE/ACM Transactions on Networking*, 2020.
- [43] S. Tschischtschek, A. Singla, and A. Krause, "Selecting sequences of items via submodular maximization," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.