Traffic-Optimal Virtual Network Function Placement and Migration in Dynamic Cloud Data Centers

Vincent Tran*, Jingsong Sun*(first two authors have equal contribution), Bin Tang*, Deng Pan†
*Department of Computer Science, California State University Dominguez Hills
vincentvtran17@gmail.com, jsung18@toromail.csudh.edu, btang@csudh.edu
†School of Computing and Information Sciences, Florida International University, pand@fiu.edu

***Abstract*—We propose a new algorithmic framework for traffic-optimal virtual network function (VNF) placement and migration for policy-preserving data centers (PPDCs). As dynamic virtual machine (VM) traffic must traverse a sequence of VNFs in PPDCs, it generates more network traffic, consumes higher bandwidth, and causes additional traffic delays than a traditional data center. We design optimal, approximation, and heuristic traffic-aware VNF placement and migration algorithms to minimize the total network traffic in the PPDC. In particular, we propose the first traffic-aware constant-factor approximation algorithm for VNF placement, a Pareto-optimal solution for VNF migration, and a suite of efficient dynamic-programming (DP)-based heuristics that further improves the approximation solution. At the core of our framework are two new graph-theoretical problems that have not been studied. Using flow characteristics found in production data centers and realistic traffic patterns, we show that a) our VNF migration techniques are effective in mitigating dynamic traffic in PPDCs, reducing the total traffic cost by up to 73%, b) our VNF placement algorithms yield traffic costs 56% to 64% smaller than those by existing techniques, and c) our VNF migration algorithms outperform the state-of-the-art VM migration algorithms by up to 63% in reducing dynamic network traffic.**

***Index Terms*—Policy-Preserving Data Centers, Dynamic Cloud Traffic, VNF Placement and Migration, Algorithms**

I. Introduction

Background and Motivation. Network Function Virtualization (NFV) is an effective technique to achieve flexible management and reduce cost in a cloud computing environment [40]. With NFV, proprietary hardware middleboxes (MBs) such as firewalls and cache proxies can now be implemented as virtual network functions (VNFs) running as lightweight containers on commodity hardware [38]. Being instantiated and deployed dynamically, VNFs provide performance and security guarantees to cloud user applications flexibly and cost-effectively. In particular, cloud operators create *service function chains* (SFCs) (or *data center policies*) that require virtual machine (VM) application traffic to traverse a sequence of VNFs to achieve aforesaid guarantees [28]. We refer to the cloud data centers that implement and enforce data center policies as *policy-preserving data centers* (PPDCs).

Fig. 1(a) shows a linear PPDC with two host servers h_1 and h_2, five switches s_1-s_5, and an SFC consisting of a firewall f_1 installed at s_1 and a cache proxy f_2 at s_2. There are two communicating VM flows: (v_1, v_1') and (v_2, v_2'), with v_1 and v_1' stored at h_1 and v_2 and v_2' at h_2. As the VM traffic between v_1 and v_1' (and traffic between v_2 and v_2') traverses f_1 and f_2 in that order, this SFC first filters out malicious traffic and then caches the content to share with other cloud users, improving both security and performance of the cloud user applications.

However, for the same reason that the VM traffic must traverse the SFC in the PPDC, it generates more network traffic, consumes more network bandwidth, and causes additional network delay among the VMs compared to traditional data centers without SFCs. For example, for the two VM pairs in Fig. 1(a), without SFC requirement, v_1 can locally communicate with v_1' (and v_2 with v_2') without incurring any network traffic and consuming any network bandwidth at all.

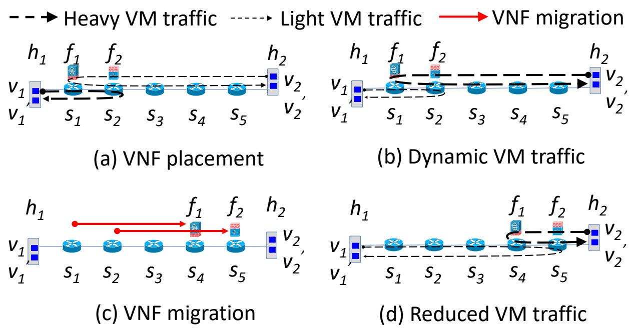


Fig. 1. An example illustrating VNF placement and migration in a PPDC.

Meanwhile, recent findings from Facebook and other production data centers [43] show that VM traffic rates (e.g., transmission rates and bandwidth demands) are highly diverse and dynamic, further exacerbating the above SFC traffic storm problem. One example is Zoom cloud conferencing [4], where one Zoom Meeting Connector VM [2] could support 200 meetings simultaneously with up to 1000 participants in a meeting. Different Zoom meetings could have a dramatically different number of participants. They could last minutes to hours with communication varied from video, voice, to text, resulting in diverse and dynamic cloud traffic. Such diverse and dynamic traffic further escalates the network traffic and increases the consumption of network resources (e.g., energy and bandwidth), as illustrated by the below example.

An Illustrative Example. Let's assume that in Fig. 1(a), the initial traffic rate of (v_1, v_1') is much larger than that of (v_2, v_2') due to diverse cloud traffic. The *traffic optimal* VNF placement is then to install f_1 on s_1 and f_2 on s_2 as shown in Fig. 1(a). This way, by traversing f_1 and f_2, the heavy traffic route of

(v_1, v_1') is "shorter" than the light traffic route of (v_2, v_2'), resulting in least amount of network traffic.

Due to dynamic traffic in PPDCs, however, if the traffic rate of (v_2, v_2') next emerges as much larger than that of (v_1, v_1'), above VNF placement is no longer traffic-optimal. As shown in Fig. 1(b), since v_2 communicates with v_2' via a route much longer than that of (v_1, v_1'), the heavy network traffic generated by (v_1, v_1') now goes through the entire PPDC and consumes much of its network bandwidth.

Our key observation is that VNFs, implemented as software, can be easily migrated to different parts of the PPDC to alleviate its dynamic network traffic adaptively. As shown in Fig. 1(c) and (d), by migrating f_1 to s_4 and f_2 to s_5, the *policy preserving* network traffic is greatly reduced as the heavy traffic of (v_2, v_2') is now confined in a short route while the light traffic of (v_1, v_1') taking a longer one.

Our Contributions. We propose a new framework of VNF placement and migration to specifically address the diverse and dynamic network traffic in the PPDCs. The framework consists of two new problems in sequence viz. TOP: *traffic-optimal VNF placement* and TOM: *traffic-optimal VNF migration*. Given a PPDC with VM flows of diverse traffic rates and an SFC that they must traverse, TOP studies how to place the VNFs inside the PPDC to minimize the total traffic incurred by policy-preserving VM communications. As this initial optimal VNF placement may become suboptimal due to dynamic traffic, TOM then adaptively migrates VNFs around the PPDC to minimize the network traffic incurred by VNF migration and VM communication. We formulate both as new graph-theoretical problems that have not been studied before. Considering VNF migration itself incurs traffic overhead and a large scale PPDC typically has hundreds of thousands of VMs with a wide range of changing traffic rates [43], both TOP and TOM are challenging problems.

In contrast, existing *traffic aware* VNF placement and migration research mostly have different objectives – While VNF placement mainly minimizes the setup cost of VNFs [16], [46], [13] or the communication cost of cloud users [56], [54], [55], [36], or maximizes the fully processed traffic [44], [45] or reliability of the SFCs [11], [41], VNF migration mostly focuses on reducing the service downtime and migration time [51], [20], [32] and resource consumption [9], [47] during the VNF migration. As such, many of above VNF placement and VNF migration problems are studied separately. By characterizing *topology-aware* costs for both VM communication and VNF migration, we are able to integrate both VNF placement and VNF migration in the same problem space to alleviate cloud network traffic and to achieve optimal utilization of cloud network resources in dynamic PPDCs.

We design optimal, approximation, and heuristic *traffic-aware* VNF placement and migration algorithms to solve TOP and TOM. In particular, we find that a special case of TOP viz. TOP-1 is equivalent to the *n-stroll problem*[1] [7], [10]. n-stroll problem finds a minimum-length path between a source and destination in a network that visits *at least* n other nodes. n-stroll is a fundamental problem that is only studied in the theory community. We are the first ones to apply it to model concrete network applications. We propose a primal-dual-based $2+\epsilon$ approximation for TOP and a Pareto-optimal solution for TOM. This is the first constant-factor approximation algorithm for the traffic-optimal VNF placement problem to the best of our knowledge. We further propose a suite of dynamic programming (DP)-based algorithms that relax the restriction implied in the approximation algorithm and empirically show that they constantly outperform the approximation algorithm. In contrast, the existing VNF placement and migration work proposed ILP solutions (which lack scalability) and heuristic algorithms (which do not have performance guarantees) [55], [34], [20], [21].

One salient feature of our framework is that it achieves ideal resource utilization for a PPDC's lifetime - after the TOP creates the initial optimal VNF placement, the TOM then executes periodically to optimize a PPDC's network resource in the face of dynamic VM traffic. Using flow characteristics found in Facebook data centers [43], we show that our VNF migration algorithms are effective in lessening dynamic traffic in PPDCs, reducing the total network traffic by 73%. Using the realistic VM traffic patterns, we show our VNF placement algorithms cost 56% to 64% smaller than existing techniques, and our VNF migration algorithms outperform the state-of-the-art VM migration algorithms by up to 63%.

II. Related Work

Traffic-Aware VNF Placement. We categorize the vast number of traffic-aware VNF placement literature into two groups. One is to optimize *resource provisioning objectives*, including minimizing the total number of VNF instances and their deployment cost [46], [13], [48], the total cost of the flow traffic [56], [16], [55], [34], the overall cloud resources (e.g., server power and network bandwidth) [54], [23], [12], [30], or achieving load-balance of cloud resources [50], [36], [49]. The other group is to achieve *admission control objectives* (where not all the requests can be satisfied) such as maximizing the total utility [33] and throughput [53] of the admitted demands, the total fully processed traffic [45], [44], or the difference between service provider's profit and the total deployment cost of VNFs [20]. As our work of minimizing total network traffic belongs to the first category, we review the related work below.

Zheng et al. [56] considered hybrid SFCs wherein different VNFs are required in the forward and backward directions between clients and servers. Cohen et al. [16] was one of the first to tackle the NFV placement problem and provided bi-criteria approximation algorithms. However, they did not consider the chain sequence in SFCs. Steering [55] was one of the first to study multiple-SFC placement and proposed a heuristic to minimize the traffic delay of all subscribers. Liu et al. [34] further formulated it as a 0-1 programming problem and showed it is NP-hard and even has no constant approximation unless P=NP. They thus proposed a few heuristics.

[1]The literatures refer to it as k-stroll problem.

In contrast with the above works, we consider one SFC at a time in the network. We are able to design the first traffic-aware constant-factor approximation VNF placement algorithm. We are aware that Tomassilli [48] et al. was the first to propose an approximation algorithm to place SFCs to minimize the total deployment cost. As the cost of deploying software VNFs is expected to be much less than the total network traffic cost in a large-scale dynamic PDDC, our work complements theirs.

Traffic-Aware VNF Migration. Huang et al. [29] studied VNF horizontal scaling (that migrates existing VNF instances) and VNF vertical scaling (that instantiates new VNF instances). They proposed ILP solutions as well as heuristics. Carpio et al. [9] used VNF replications to load-balance and reduce VNF migrations. Yi et al. [52] proposed to migrate VNFs from poor-state nodes to good-state nodes to maintain the network balance and performance. A few works targeted dynamic cloud traffic. Cho et al. [14] used VNF migration to achieve lower network latency considering dynamic resource availability. However, it did not consider migration costs. Liu et al. [35] maximized the service provider's profit by considering that existing users can move around and new users can join in.

There are a few works that studied VNF placement and migration in the same context [20], [21], [18], [26]. In [20], the objective of VNF placement is to minimize the rejected SFC bandwidth, and the aim of VNF migration is to consolidate VNF instances to reduce the server energy consumption and QoS degradation. Farkiani et al. [21] considered chains with different priorities and studied the chain deployment and reconfiguration to maximize the service provider's profit. Like most research, both works proposed ILP solutions (which lack scalability) as well as heuristic algorithms (which do not have performance guarantees). Cziva et al. [18] studied dynamic and latency-optimal VNF placement at network edge considering changing network dynamics. They proposed ILP formulation and used the theory of optimal stopping to decide when to migrate VNFs. Our work differs from them in both objective and solution techniques. We focus on traffic mitigation in large-scale dynamic cloud data centers and propose the first constant-factor efficient approximation algorithm for VNF placement and a Pareto-optimal VNF migration.

III. Preliminaries

System Model. We model a PPDC as an undirected and weighted graph $G(V, E)$ where $V = V_h \cup V_s$ includes a set of hosts $V_h = \{h_1, h_2, ..., h_{|V_h|}\}$ and a set of switches $V_s = \{s_1, s_2, ..., s_{|V_s|}\}$. E is a set of edges, each connecting either one switch to another or a switch to a host. Fig. 2 shows a $k = 4$ fat-tree [6] PPDC where k is the number of ports of a switch.[2] We assume that each switch is attached with a server that can implement various VNFs [55]. There is an SFC consisting of n VNFs $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ that need to be placed inside the PPDC. Fig. 2 shows three VNFs f_1, f_2 and f_3 installed on attached servers of different switches in the PPDC.[3] As a switch and its attached server are connected by high-speed optical fibers, the delay between them is negligible compared to that among switches and hosts [27]. We assume policy consistency maintenance mechanisms (i.e., FlowTags [22]) are available for VM traffic redirection during VNF migration.

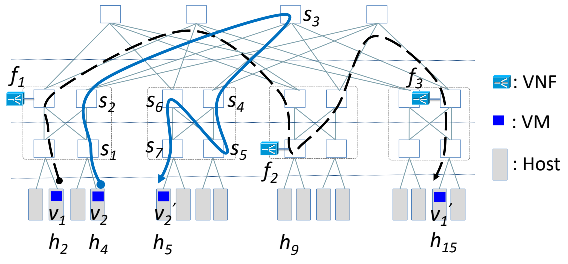


Fig. 2. A PPDC with 16 hosts: h_1, h_2, ..., and h_{16}, 3 VNFs: f_1, f_2, and f_3, and two VM flows: (v_1, v_1') and (v_2, v_2'). The policy-preserving communication for (v_1, v_1') is shown in black dashed line. The optimal 7-stroll for (v_2, v_2') is shown in solid blue line, as explained in Example 3.

As the east-west traffic accounts more than 70 percent of the traffic in a data center [1], [3], and most east-west cloud traffic is pairwise [39], we focus on pairwise VM communication. We assume that there are l pairs of communicating VMs $\mathcal{P} = \{(v_1, v_1'), (v_2, v_2'), ..., (v_l, v_l')\}$ already placed on the hosts, where $v \in \mathcal{V} = \{v_1, v_1', v_2, v_2', ..., v_l, v_l'\}$ is placed at host $s(v)$ and all the VMs have the same size. For any VM flow (v_i, v_i'), v_i and v_i' are referred to as its *source* and *destination VM* and $s(v_i)$ and $s(v_i')$ as its *source* and *destination host* respectively. We denote the traffic rate of (v_i, v_i'), which could be the communication frequency or bandwidth demand between v_i and v_i', as λ_i and the *traffic rate vector* as $\overrightarrow{\lambda} = \langle \lambda_1, \lambda_2, ..., \lambda_l \rangle$. In Fig. 2, there are two VM flows: (v_1, v_1') and (v_2, v_2') with $\overrightarrow{\lambda} = \langle 1, 100 \rangle$. Note that as the VM traffic rates change over time in a dynamic PPDC, $\overrightarrow{\lambda}$ is not a constant vector. Considering that network links are generally provisioned around 40% of utilization to protect against failures and packet loss [31], we assume there are enough edge bandwidths. Table I shows all the notations.

Topology-Aware Cost Model. Each edge $(u, v) \in E$ has a weight $w(u, v)$, indicating either the network delay or energy cost on this edge caused by one unit of VM communication or VNF migration. Given any host or switch u and v, let $c(u, v)$ denote the total cost from u to v (i.e., the sum of the weights of all the edges traversed by VM communication or VNF migration from u to v). Thus the *communication cost* of any VM flow (v_i, v_i') is $\lambda_i \cdot c\big(s(v_i), s(v_i')\big)$ and the *migration cost* of migrating any VNF in $\mathcal{F}$ from switch u to switch v is $\mu \cdot c(u, v)$. Here μ is *VNF migration coefficient*, which is defined as the ratio between costs of VNF migration and VM communication. It represents the relative size of memory

[2]We use fat-trees for illustration purpose. However, the problems and solutions apply to any data center topology.

[3]As the attached server of each switch has limited resources thus can install a limited number of VNFs [13], we assume that different VNFs of an SFC are installed on servers attached on different switches, but a VNF can be installed on any of the switches.

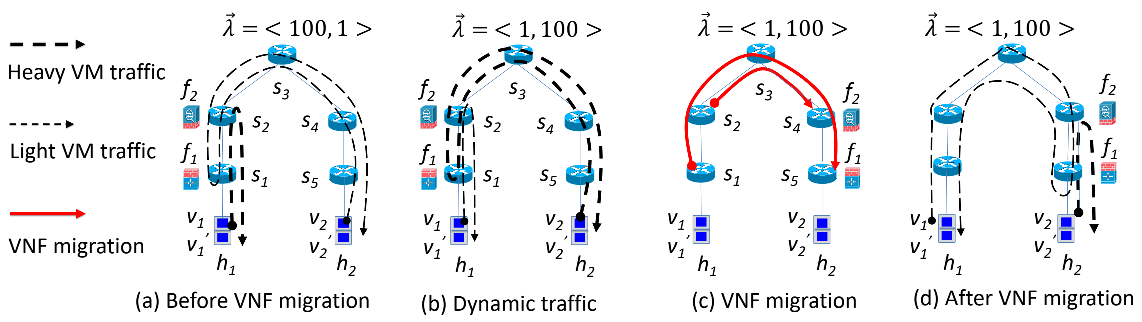


Fig. 3. VNF migration achieves 58.6% of total cost reduction in a k=2 PPDC.

or data packet transferred in VNF migration and VM communication. We will quantify μ in Section VI. Our topology-aware model is different from the well-known *pre-copy* VM migration model [37], wherein the cost of migrating a VM or a VNF is measured by the hypervisor of its host and does not take into account the network topology. Instead, our model captures network traffic incurred in VM communication and VNF migration and is more suitable to study VNF placement and migration in a large-scale dynamic cloud data center.

Service Function Chainings (SFCs). As an SFC consists of multiple VNFs of different security and performance functions, we assume one SFC is sufficient to provide security and performance guarantees to cloud applications [3]. An SFC, denoted as $(f_1, f_2, ..., f_n)$, requires that the VM traffic to go through VNFs f_1, f_2, ..., and f_n in that specific order. We refer to f_1 (and f_n) as *ingress* (and *egress*) VNF, and the switch where the ingress (and egress) VNF is installed as *ingress* (and *egress*) switch. In Fig. 2, (v_1, v_1') traverses SFC (f_1, f_2, f_3), resulting in communication cost of $1 \times 10 = 10$ (black dashed line). Here we use unweighted costs (i.e., number of edges) only for the purpose of illustration. Next we quantify the benefit of using VNF migration to reduce network cost.

EXAMPLE 1: Fig. 3 shows a k=2 fat tree PPDC with two hosts h_1 and h_2. There are two VM flows (v_1, v_1') and (v_2, v_2'), with v_1 and v_1' at h_1 while v_2 and $v_2{}'$ at h_2. $\overrightarrow{\lambda} = \langle 100, 1\rangle$ and $\mu = 1$. There are two VNFs f_1 and f_2. Fig. 3(a) shows one initial optimal VNF placement, where f_1 is installed on switch s_1 and f_2 on switch s_2, resulting in total communication cost of $100 \times 4 + 1 \times 10 = 410$ (shown in black dashed lines). However, due to dynamic traffic, $\overrightarrow{\lambda}$ next changes to $\langle 1, 100\rangle$, as shown in Fig. 3(b). This results in a dramatic increase of total communication cost to $1 \times 4 + 100 \times 10 = 1004$. Our solution to reduce the cost is to migrate f_1 to s_5 and f_2 to s_4, shown in solid red line in Fig. 3(c). Although this incurs migration cost of 6 for VNF migration, the total cost of the VM communication (shown in Fig. 3(d)) reduces to $1 \times 10 + 100 \times 4 = 410$, a 58.6% of total cost reduction. This fat tree PPDC is indeed the same linear PPDC in Fig. 1.

TABLE I
NOTATION SUMMARY

Notation	Description
$G(V, E)$	A PPDC graph, where $V = V_h \cup V_s$
$w(u, v)$	Weight of an edge $(u, v) \in E$
V_h	$V_h = \{h_1, h_2, ..., h_{\lvert V_h \rvert}\}$ is the set of $\lvert V_h\rvert$ hosts
V_s	$V_s = \{s_1, s_2, ..., s_{\lvert V_s \rvert}\}$ is the set of $\lvert V_s\rvert$ switches
$\mathcal{F}$	$\mathcal{F} = \{f_1, f_2, ..., f_n\}$ is the set of n VNFs
$\mathcal{P}$	$\mathcal{P} = \{(v_i, v_i'), ..., (v_l, v_l')\}$ is the set of l VM flows
$\mathcal{V}$	$\mathcal{V} = \{v_1, ..., v_l, v_1', ..., v_l'\}$
$s(v)$	The host in V_h where VM $v \in \mathcal{V}$ is stored
λ_i	Traffic rate between v_i and v_i', $1 \le i \le l$
$\overrightarrow{\lambda}$	$\overrightarrow{\lambda} = \langle \lambda_1, \lambda_2, ..., \lambda_l\rangle$ is the changing traffic rate vector
$c(u, v)$	Cost between hosts (or switches) u and v
$p(j)$	VNF placement function p, f_j is placed at switch $p(j)$
$C_a(p)$	Total VM communication cost with VNF placement p
μ	VNF migration coefficient
$m(j)$	VNF migration function m, f_j is migrated to switch $m(j)$
$C_b(p, m)$	Total VNF migration cost of migrating from p to m
$C_a(m)$	Total VM communication cost after migrating from p to m
$C_t(p, m)$	Total VNF migration and VM communication cost after migrating from p to m; $C_t(p, m) = C_b(p, m) + C_a(m)$

IV. TOP: Traffic-Optimal VNF Placement

In this section, we formulate the TOP and prove its NP-hardness. Then we focus on a special case of TOP with one VM pair viz. TOP-1, and design a primal-dual-based approximation algorithm. We then relax an assumption made in the approximation algorithm, propose a DP-based heuristic algorithm and give its optimality condition. Finally, we solve the TOP based on our DP solution for TOP-1.

1) Problem Formulation: We define a *VNF placement function* as $p : \mathcal{F} \to V_s$, which places VNF $f_j \in \mathcal{F}$ at switch $p(j) \in V_s$. Given any VNF placement p, denote the *total communication cost* of all the l VM flows under p as $C_a(p)$.

$$\begin{aligned} C_a(p) = &\sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(p(j), p(j+1)\big) + \\ &\sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(s(v_i), p(1)\big) + c\big(p(n), s(v_i')\big)\Big). \end{aligned} \tag{1}$$

The objective of TOP is to find a p to minimize $C_a(p)$. Note that for any VM flow, the ingress switch is always $p(1)$, and the egress switch is always $p(n)$. TOP generalizes the classic *p-median problem* [42]. The p-median problem places p facilities in a network while minimizing the total distance between demand nodes and their closest facilities. TOP, however, not only identifies p locations to install the VNFs, but for each flow, it needs to traverse all the p VNFs in some order instead of just accessing the closest VNF. Before solving TOP, we first focus on a special case of TOP with only one VM flow (v_1, v_1'). We refer to it as TOP-1. Below we prove that TOP-1 is equivalent to *n-stroll problem* [10], [7], which is NP-hard.

n-stroll problem. Given a weighted graph G=(V, E) with nonnegative length w_e on edge $e \in E$, two special nodes s and t, and an integer n, n-stroll problem finds an s-t path or walk (i.e., a stroll) of minimum length that visits at least n distinct nodes excluding s and t. When s=t, it is called *n-tour problem*. The triangle inequality holds for all edges: for $(x, y), (y, z), (z, x) \in E$, $w(x, y) + w(y, z) \geq w(z, x)$. Fig. 4(a) shows an optimal 2-stroll between s and t: s, D, t, C, and t, which is a walk with a cost of 6. The other 2-stroll s, A, B, and t is a path with cost of 7, thus is not optimal.

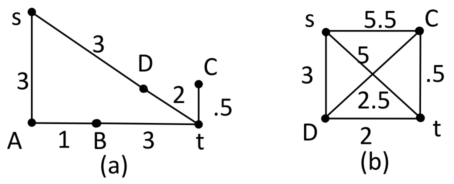


Fig. 4. Illustrating (a) n-stroll problem and (b) how DP in Algo. 2 works.

Theorem 1: TOP-1 is equivalent to the n-stroll problem.
Proof: To prove that they are equivalent, we first reduce any instance of TOP-1 to an instance of n-stroll in polynomial time and then do the reverse.

First, given any instance of TOP-1 with a PPDC graph $G(V = \{V_h \cup V_s\}, E)$, where the only pair of VMs v_1 and v_1' are located at hosts $s(v_1), s(v_1') \in V_h$ respectively, we construct an induced new graph $G'(V', E')$ where $V' = V_s \cup \{s(v_1), s(v_1')\}$ and $E' = \{(u, v) \in E | u, v \in V'\}$. We claim that a placement of n VNFs for (v_1, v_1') in G is optimal if and only if the corresponding n-stroll in G' that starts at $s(v_1)$ and ends at $s(v_1')$ is optimal. As a VNF placement gives minimum cost for (v_1, v_1') in G and each of the n VNFs is placed on a different switch, the resulted $s(v_1)$-$s(v_1')$ stroll in G' thus visits these n distinct nodes with minimum cost. On the other hand, as an optimal n-stroll in G' must visit at least n distinct other nodes between $s(v_1)$ to $s(v_1')$, we place f_1, f_2, ..., and f_n at the first n distinct nodes respectively. When v_1 communicate with v_2 by traversing f_1, f_2..., and f_n in that order, it gives (v_1, v_1') minimum communication cost in G, as the $s(v_1)$-$s(v_1')$ stroll is a minimum n-stroll in G'.

As a VNF placement gives minimum cost for (v_1, v_1') in G and each of the n VNFs is placed on a different switch, the resulted $s(v_1)$-$s(v_1')$ stroll in G' thus visits these n distinct nodes with minimum cost. On the other hand, as an optimal n-stroll in G' must visit at least n distinct other nodes between $s(v_1)$ to $s(v_1')$, we place f_1, f_2, ..., and f_n at the first n distinct nodes respectively. When v_1 communicate with v_2 by traversing f_1, f_2..., and f_n in that order, it gives (v_1, v_1') minimum communication cost in G, as the $s(v_1)$-$s(v_1')$ stroll is a minimum n-stroll in G'.

Fig. 5 (a) and (b) show the instances of G and G' with one VM flow (v_1, v_1') in Fig. 3. As $s(v_1) = s(v_1') = h_1$ and one optimal VNF placement for (v_1, v_1') in G is f_1 at s_1 and f_2 at s_2, the optimal s-t 2-tour in G' is s, s_1, s_2, s_1, and t. ∎

2) Algorithms for TOP-1: Inspired by [10], Algo. 1 below is a primal-dual based approximation algorithm for TOP-1.

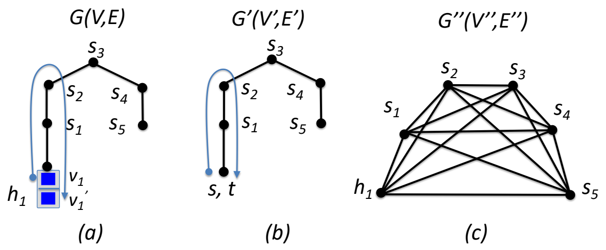


Fig. 5. Proving TOP-1 is equivalent to n-stroll problem.

We first give the primal ILP of TOP-1.

Given $G(V, E)$, $V = V_s \cup \{s(v_1), s(v_1')\}$, let x_v indicate if $v \in V_s$ is selected to place a VNF, and y_e denote if an edge $e \in E$ is on the path. Let $\delta(S)$ denote the set of edges with exactly one endpoint in set S. The primal ILP of TOP-1 is:

$$\min \lambda_1 \cdot \sum_{e \in E} c_e \times y_e \tag{2}$$

s.t.

$$x_v = \{0, 1\}, \quad \forall v \in V_s \tag{3}$$

$$y_e = \{0, 1\}, \quad \forall e \in E \tag{4}$$

$$\sum_{e \in \delta(U)} y_e \geq 1, \quad \forall U \subseteq V, s(v_1') \in U, s(v_1) \notin U \tag{5}$$

$$\sum_{e \in \delta(S)} \geq 2 \cdot x_v, \quad \forall S \subseteq V_s, \forall v \in S \tag{6}$$

$$\sum_{v \in V_s} x_v \geq n, \quad \forall v \in V_s. \tag{7}$$

Constraints 5 and 6 construct a path between hosts $s(v_1)$ and $s(v_1')$ by selecting an edge in every cut separating them, and Constraint 7 guarantees this path has least n switches.

Given above ILP of the TOP-1 and its feasible dual solution y, Algo. 1 iteratively finds a feasible primal solution x that obeys the complementary slackness conditions [5] w.r.t. y.

Algorithm 1: An Approximation Algorithm for TOP-1.

Step 1. It considers the LP-relaxation of above ILP (i.e., $0 \leq x_v \leq 1$ and $0 \leq y_e \leq 1$), and then relaxes the complementary slackness condition related to its dual variables.

Step 2. It iteratively adds edges, paying for them with increases to variables in the dual (*growth* phase), and then deletes edges to obtain the final path that spans n switches (*pruning* phase). On the pruned tree constructed, it starts with $s(v_1)$ and ends at $s(v_1')$ while traversing all the edges in the tree at most twice. This gives the n-stroll for $s(v)$ and $s(v_1')$.

Algo. 1 takes $O(|V|^5 \cdot \log|V|)$ [10]. We give Theorem 2 without proof that it is a $2 + \epsilon$ approximation for TOP-1 [10].

Theorem 2: In Algo. 1, the primal problem of TOP-1 costs no more than $2+\epsilon$ times the value of the feasible dual solution. This implies that the cost of the resulted n-stroll is with a factor of $2 + \epsilon$ of the optimal.

Discussions. In Algo. 1, as the ILP counts the weight of each edge on the n-stroll once, it implicitly assumes that the optimal n-stroll must be a path that visits each edge once. However, Fig. 4(a) shows that an optimal n-stroll can be a walk that visits an edge multiple times. Thus, this is a strong assumption.

Besides, to achieve rigorous analysis of its performance bound, Algo. 1 proposes complicated procedures (e.g., Step 2) that cannot be easily implemented for large-scale PPDCs.

DP Algorithm for TOP-1. We thus propose a more practical and time-efficient VNF placement heuristic viz. Algo. 2 for TOP-1. Our key observation is that although finding a shortest s-t stroll visiting n distinct nodes is NP-hard, finding one visiting n edges (not necessarily distinct) can be solved optimally and efficiently using DP. Algo. 2 takes as input a complete graph $G''(V'', E'')$ that is transformed from the PPDC graph $G(V, E)$ as follows. $V'' = \{s(v_1), s(v_1')\} \cup V_s$; for an edge $(u, v) \in E''$, its cost $c_{(u,v)}$ is $\lambda_1 \cdot c(u, v)$, the communication cost of (v_1, v_1') between u and v in G. As there does not always exist an $s(v_1)$-$s(v_1')$ stroll of exactly $n+1$ edges in G while there always exists one in G'' (as long there are at least $n+1$ edges in G''), using G'' overcomes an obstacle otherwise faced by using G.

Algo. 2 finds a shortest $s(v_1)$-$s(v_1')$ stroll with $n+1$ edges (lines 4-10) and checks if it traverses n distinct switches (lines 11-19). If not, it finds a stroll with $n+2$ edges, so on and so forth, until n distinct switches are found (lines 20-21). It finally places f_1, ..., f_n on the first n switches and return the cost of the n-stroll (lines 23-25). Its time complexity is $O(n \cdot |V|^4)$. Note that Algo. 2 also works for n-tour problem where $s(v_1)$=$s(v_1')$ and the special case that n distinct switches are already on the shortest path between $s(v_1)$ and $s(v_1')$.

Algorithm 2: A DP Algorithm for TOP-1.

Input: A complete graph $G''(V'', E'')$, $s(v_1)$, $s(v_1')$, and an SFC $(f_1, f_2, ..., f_n)$.

Output: $stroll(G'', s(v_1), s(v_1'), n, p)$, cost of an $s(v_1)$-$s(v_1')$ stroll in G'' visiting at least n distinct switches.

Notations: e: index for edges; i: index for switches;
$c(u, s(v_1'), e)$, $successor(u, s(v_1'), e)$: cost and u's successor in a u-$s(v_1')$ stroll with e edges, $+\infty$ and -1 initially;
r: num. of edges needed on $s(v_1)$-$s(v_1')$ stroll, initially $n+1$;
p: an array storing distinct switches on $s(v_1)$-$s(v_1')$ stroll;
num: the number of distinct switches in p;
$found$: true if it has found a $s(v_1)$-$s(v_1')$ stroll with at least n distinct switches; initially false;

```
1. $V'' = \{u_1, ..., u_{|V''|}\}$, let $u_a = s(v_1)$ and $u_{|V''|} = s(v_1')$;
2. $\forall u_i, u_j \in V''$ with $i \neq j$, $c(u_i, u_j, 1) = c_{u_i,u_j}$,
   $successor(u_i, u_j, 1) = u_j$, $successor(u_j, u_i, 1) = u_i$;
   $\forall u_i \in V''$, $c(u_i, u_i, 1) = +\infty$, $successor(u_i, u_i, 1) = -1$;
3. while $(\neg found)$
4.     for $(e = 2; e <= r; e{+}{+})$      // edges in $u_i$-$s(v_1')$ stroll
5.         for $(i = 1; i \leq |V''| - 1; i{+}{+})$      // node $u_i$
6.             for (each $u$, $u \neq u_i \wedge u \neq s(v_1') \wedge$
                   $u_i \neq successor(u, s(v_1'), e-1)$)
7.                 if $\big(c(u_i, s(v_1'), e) > c_{u_i,u} + c(u, s(v_1'), e-1)\big)$
8.                     $c(u_i, s(v_1'), e) = c_{u_i,u} + c(u, s(v_1'), e-1)$;
9.                     $successor(u_i, s(v_1'), e) = u$;
10.                end if;
11.    $num = 0$; $p = \phi$ (empty set), $e{-}{-}$;
12.    $b = successor(s(v_1), s(v_1'), e)$;
13.    while $(e > 1)$
14.        if $(b \neq s(v_1) \wedge b \neq s(v_1') \wedge b \notin p)$
15.            $p[num] = b; num{+}{+}$;
16.        end if;
17.        $e{+}{+}$;
18.        $b = successor(b, s(v_1'), e)$;
19.    end while;
20.    if $(num < n)$ $r{+}{+}$; // less than $n$ distinct switches
21.    else $found$ = true;
22. end while;
23. Place $f_1$, ..., $f_n$ on the first $n$ switches stored in $p$;
24. $stroll(G'', s(v_1), s(v_1'), n, p) = c(s(v_1), s(v_1'), r)$;
25. RETURN $stroll(G'', s(v_1), s(v_1'), n, p)$.
```

EXAMPLE 2: The complete graph of Fig. 4(a) is partially shown in Fig. 4(b) (other edges are irrelevant for discussion thus not shown). To find the optimal s-t 2-stroll in Fig. 4(a), Algo. 2 instead finds the optimal s-t 3-edge stroll on Fig. 4(b). In particular, it finds C-t 1-edge stroll, D-t 2-edge stroll, and s-t 3-edge stroll in that order, where the s-t 3-edge stroll s, D, C, and t gives optimal cost of 6. Note that if Algo. 2 takes non-complete graph in Fig. 4(a) as input, it finds a 3-edge stroll s, A, B, t of cost 7, which is not optimal. □

One obstacle of the DP approach is that an edge can be traversed multiple times, incurring more cost but not finding more distinct nodes. Algo. 2 overcomes it partially by making sure that the same edge is not traversed twice consecutively (line 6). We illustrate it by below example.

EXAMPLE 3: Fig. 2 shows a VM flow (v_2, v_2') with v_2 and v_2' placed at h_4 and h_5 respectively. To place 7 VNFs between v_2 and v_2' is to find a 7-stroll between h_4 and h_5. Algo. 2 finds an 8-edge path traversing 7 distinct switches (shown in solid blue line): h_4, s_1, s_2, s_3, s_4, s_5, s_6, s_7, and h_5. There is an 8-edge walk between h_4 and h_5 that traverses 5 distinct switches: h_4, s_1, s_2, s_1, s_2, s_3, s_4, s_7, and h_5, which is not selected by Algo. 2 due to loops between s_1 and s_2. □

Theorem 3 below gives the sufficient condition for the optimality of Algo. 2.

Theorem 3: Let $successor_i$ be the i^{th} switch on the $s(v_1)$-$s(v_1')$ stroll found in Algo. 2, $1 \leq i \leq n$. If the $(n+1-i)$-edge stroll found in Algo. 2 that starts at $successor_i$ and ends at $s(v_1')$ has the minimum cost among all the $(n+1-i)$-edge strolls that end at $s(v_1')$, then Algo. 2 is optimal for TOP-1.

Proof. We show that both *overlapping subproblem* and *optimal substructure* needed for the optimality of a DP are satisfied. First, given any instance of TOP-1, as $successor_i$ is the i^{th} switch on the $s(v_1)$-$s(v_1')$ stroll found in Algo. 2, the original $(n+1)$-edge $s(v_1)$-$s(v_1')$ stroll problem encompasses it n subproblems viz. $successor_i$-$s(v_1')$ $(n+1-i)$-edge stroll problem, $1 \leq i \leq n$. This satisfies the overlapping subproblems. Second, consider all the subproblems, as each $successor_i$-$s(v_1')$ $(n+1-i)$-edge stroll found by Algo. 2 has the minimum cost among all the $(n+1-i)$-edge stroll that ends at $s(v_1')$, it satisfies the optimal substructure. ∎

For example, the optimal s-t 3-edge stroll found in Fig. 4(b) s, D, C, and t satisfies this sufficient condition, as the solutions to its the subproblems viz. C-t 1-edge stroll, D-t 2-edge stroll, and s-t 3-edge stroll are all optimal.

By taking the complete graph as input and avoiding the edge loops, Algo. 2 dramatically improves the efficiency of searching for distinct nodes in TOP-1. We show in experiments that Algo. 2 constantly outperforms the performance guarantee of $2 + \epsilon$ provided by the approximation Algo.1 and performs very close to the optimal algorithm viz. Algo. 4 presented next.

3) Algorithms for TOP: For TOP with $l > 1$, as there exists simple solutions for cases of $n = 1, 2$, we only consider $n \geq 3$. Algo. 3 checks all pairs of ingress and egress switches $p(1)$ and $p(n)$. For each pair, it finds the rest n-2 switches by solving an $(n$-2$)$-stroll problem with $s = p(1)$ and $t = p(n)$ using Algo. 2. Finally it juxtaposes them to find an n-stroll that gives the minimum cost. Its time complexity is $O(n \cdot |V|^6)$.

```
Algorithm 3: VNF Placement Algorithm for TOP.
Input: A PPDC graph $G(V,E)$ with VM placement $s(v)$,
       $v \in \mathcal{V}$, and an SFC $(f_1, f_2, ..., f_n)$, $n \geq 3$.
Output: A VNF placement $p$, total VM comm. cost $C_a(p)$.
Notations: $x$: set of switches storing $f_2, f_3, ..., f_{n-1}$;
1.   Compute $G''(V'' = V, E'')$, $C_a(p) = +\infty$;
2.   for $(1 \leq i < |V_s|)$
3.     for $(i + 1 \leq j \leq |V_s|)$
4.       $a = \sum_{k=1}^{l} \lambda_k \cdot \big(c(s(v_k), s_i) + c(s_j, s(v'_k))\big)$;
5.       $b = stroll(G'', s_i, s_j, n-2, x)$; // Call Algo. 2
6.       if $\big((a + b) \leq C_a(p)\big)$
7.         $C_a(p) = a + b$, $p(1) = s_i$, $p(n) = s_j$;
8.         for $(2 \leq i \leq n-1)$ $p(i) = x[i-2]$;
9.     end for;
10.  end for;
11.  RETURN $p$ and $C_a(p)$.
```

Below we present Algo. 4, an exhaustive algorithm that solves TOP optimally. It takes $O(|V|^n)$. Although it is not time-efficient, as it can be implemented easily, we compare it with other algorithms as benchmark.

```
Algorithm 4: Exhaustive VNF Placement for TOP.
Input: A PPDC $G(V,E)$ with VM placement $s(v)$, $v \in \mathcal{V}$,
       and an SFC $(f_1, f_2, ..., f_n)$.
Output: A VNF placement $p$ and the total cost $C_a(p)$.
1. $C_a(p) = +\infty$;
2. Among all $|V_s| \cdot (|V_s| - 1) \cdot ..., \cdot (|V_s| - n + 1)$ VNF
     placements, find $p$ that gives the minimum cost $C_a(p)$.
3. RETURN $p$ and $C_a(p)$.
```

V. TOM: Traffic-Optimal VNF Migration

In this section, we first formulate the TOM and show its NP-hardness. Before presenting our traffic-optimal VNF migration algorithm, we introduce *VNF migration frontier*, a key concept used in our algorithm. We then show that the VNF migration frontier represents a Pareto front, which is a set of optimal solutions for multi-objective optimization problems (MOOPs). Finally, we give a sufficient condition for the optimality of our VNF migration algorithm.

Let's represent the initial VNF placement in TOM as $p : \mathcal{F} \to V_s$, indicating that $f_j \in \mathcal{F}$ is located at switch $p(j) \in V_s$. Note p is computed by various VNF placement algorithms (i.e., Algo. 1, 2, 3, and 4) according to the traffic rates at that moment. The total communication cost of all the l VM pairs is thus $C_a(p)$ (Eq. 1). However, as the traffic rate vector $\overrightarrow{\lambda}$ changes over time due to dynamic traffic, the $C_a(p)$ is no longer optimal, necessitating VNF migration.

1) Problem Formulation: We define a VNF *migration function* as $m : \mathcal{F} \to V_s$, meaning that $f_j \in \mathcal{F}$ will be migrated from switch $p(j)$ to switch $m(j)$ ($m(j) = p(j)$ if f_j does not migrate). Let $C_b(p, m) = \mu \cdot \sum_{j=1}^{n} c\big(p(j), m(j)\big)$ be the *total migration cost* of migrating all the n VNFs from p to m and $C_a(m)$ be the *total communication cost* of all VM flows *after* migrating to m. Let $C_t(p, m) = C_b(p, m) + C_a(m) =$

$$\mu \cdot \sum_{j=1}^{n} c\big(p(j), m(j)\big) + \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(m(j), m(j+1)\big)$$
$$+ \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(s(v_i), m(1)\big) + c\big(m(n), s(v'_i)\big)\Big). \tag{8}$$

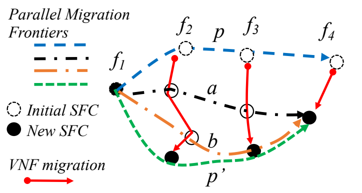
The objective of TOM is to find a VNF migration m that minimizes $C_t(p, m)$. Note that new users join for the first time [35] is a special case of TOM, wherein their traffic rates change from zero to some positive values. Below we show that TOP is a special case of TOM, thus TOM is also NP-hard.

Theorem 4: TOP is a special case of TOM with $\mu = 0$.

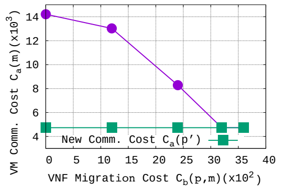
Proof: Plug $\mu = 0$ into Eq. 8, we get $C_t(p, m) = \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(m(j), m(j+1)\big) + \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(s(v_i), m(1)\big) + c\big(m(n), s(v'_i)\big)\Big)$. As TOP is to find a VNF placement, we replace m with p in r.h.s. of above equation and get $C_t(p, m) = \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(p(j), p(j+1)\big) + \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(s(v_i), p(1)\big) + c\big(p(n), s(v'_i)\big)\Big) \overset{\text{Eq. 1}}{=} C_a(p)$. ∎

2) Algorithms for TOM: To design VNF migration algorithms, we first compute new VNF placement p' using Algo. 3 and we have $C_a(p') \leq C_a(p)$. To reduce VM communication cost, we migrate VNF f_j from $p(j)$ towards $p'(j)$ gradually along the shortest path between them. As $p(j)$ incurs zero VNF migration cost while $p'(j)$ incurs minimum VM communication cost, the challenge is to decide how far each VNF migrates. We give the below definition.

Definition 1: (VNF Migration Frontiers.) Denote the shortest path between $p(j)$ and $p'(j)$ as S_j. Let $h_j \geq 1$ be the number of switches on S_j ($h_j = 1$ if $p(j) = p'(j)$), and $h_{max} = \max\{h_j\}$. The *VNF migration frontiers* $\mathcal{F}$ are sets of n switches, each of which is from a different S_j. That is, $\mathcal{F} = \{\langle s_{k_1}, s_{k_2}, ..., s_{k_n}\rangle | s_{k_j} \in S_j, 1 \leq j \leq n\}$. □



(a) VNF parallel migration frontiers.



(b) Pareto front, k=16, n=6.

Fig. 6. Illustrating VNF migration Algo. 5.

$\mathcal{F}$ essentially represents all $|\mathcal{F}| = \prod_{j=1}^{n} h_j$ possible schemes of migrating VNFs from p to p'. Our VNF migration algorithm (Algo. 5) is thus to find a migration frontier that gives the minimum total cost of VNF migration and VM communication. As $|\mathcal{F}|$ becomes large in a large scale PPDC and it takes time to find the best migration scheme, our algorithm focuses on *parallel migration frontiers* defined below.

Definition 2: (**VNF Parallel Migration Frontiers.**) Denote the k^{th} switch in S_j as $s_{j,k}$, $1 \leq k \leq h_j$; $s_{j,1} = p(j)$ and $s_{j,h_j} = p'(j)$. Let $\mathbb{P} = [p_{i,j}]$ be an $(h_{max} \times n)$ matrix where $p_{i,j} = s_{j,i}$ if $i \leq h_j$ and $p_{i,j} = s_{j,h_j}$ otherwise. The h_{max} rows in $\mathbb{P}$ are the h_{max} VNF parallel frontiers. □

EXAMPLE 4: Fig. 6(a) illustrates how parallel VNF migration frontiers look like. It shows the migration paths of VNFs using red solid lines. It also shows the initial SFC placement p and the new SFC placement p'. As $h_1 = 1$, $h_2 = 4$, $h_3 = 3$, and $h_4 = 2$, there are $|\mathcal{F}| = 24$ VNF migration frontiers. Four of them are VNF parallel frontiers viz. p, a, b, and p', represented by different dashed lines. □

With the above preparation, we present our VNF migration algorithm viz. Algo. 5 below. It first calls Algo. 3 to compute the new VNF placement p' (lines 1-3). Next, it finds all parallel frontiers between p' and initial VNF placement p. Among them, it finds m, the parallel frontier with the minimum total cost (lines 4-19). Finally, it migrates VNF f_j from switch $p(j)$ to switch $m(j)$ along the shortest path between them and returns the total cost (lines 20 and 22). Its time complexity is $O(|V|^3 + n \cdot D)$, where D is the diameter (i.e., the greatest distance between any pair of vertices) of G.

```
Algorithm 5: VNF Migration Algorithm for TOM.
Input: A PPDC graph $G(V,E)$ with VM pair placement
       $s(v)$, VNF placement $p(j)$, $\mu$, and $\overrightarrow{\lambda}$;
Output: A VNF migration $m$ and its total cost $C_t(p,m)$.
Notations: $fr$: the current parallel migration frontier;
  $i$: index of frontiers; $j$: index of VNFs;
  $fr[j]$: the switch where VNF $f_j$ is located in $fr$;
  $fr|\{j,a\} = \langle p(1),...,p(j-1),a,p(j+1),...,p(n)\rangle$:
     $fr$ with its $j^{th}$ switch replaced by switch $a$;
1.   Compute new VNF placement $p'$ using Algo. 3;
     Let $h_j$ be the number of switches on the shortest path
        between $p(j)$ and $p'(j)$, and $h_{max} = max\{h_j\}$;
     Let $s_{j,k}$ be the $k^{th}$ switch from $p(j)$ to $p'(j)$;
2.   $m = fr = \langle p(1), p(2), ..., p(n)\rangle$; // initial frontier
3.   $C_b(p,m) = 0$; $C_t(p,m) = C_a(m)$;
4.   if ($h_{max} \geq 2$) // if $p$ and $p'$ are not the same
5.       for ($1 \leq j \leq n$) $h_j$++;
6.       end for;
7.       for ($2 \leq i \leq h_{max}$) // $i^{th}$ parallel frontier
8.           for ($1 \leq j \leq n$) // VNFs
9.               if ($h_j \geq 1$) // a new switch to migrate to
10.                  $C_b(p,fr)$ += $\mu \cdot c(fr[j], s_{j,i})$;
11.                  $fr = fr|\{j, s_{j,i}\}$;
12.                  $h_j$++;
13.              end if;
14.          end for;
15.          $C_t(p,fr) = C_a(fr) + C_b(p,fr)$;
16.          if $(C_t(p,fr) < C_t(p,m))$
17.              $C_t(p,m) = C_t(p,fr)$; $m = fr$;
18.          end if;
19.      end for;
20.      Migrates each $f_j$ from switch $p(j)$ to switch $m(j)$.
21.  end if;
22.  RETURN $m$ and $C_t(p,m)$.
```

Discussions. As migrating VNFs from p to p' decreases the VM communication cost $C_a(m)$ at the price of increasing the VNF migration cost $C_b(p,m)$, Algo. 5 attempts to achieve a balance between $C_a(m)$ and $C_b(p,m)$. We conjecture that its solution is a *Pareto-optimal* point, where neither $C_a(m)$ nor $C_b(p,m)$ can be improved (i..e, decreased) without deteriorating (i.e, increasing) the other. As the goal of TOM is to minimize $C_t(p,m) = C_a(m) + C_b(p,m)$, we can treat it as a multi-objective optimization problem (MOOP) [15] that tries to minimize $C_a(m)$ and $C_b(p,m)$ simultaneously. As Pareto front [15] consisting of Pareto-optimal points is an "optimal" solution of MOOPs, we empirically investigate if VNF migration frontier found in Algo. 5 yields a Pareto front.

Fig. 6(b) considers a k=16 fat tree PPDC with n=6 VNFs and migration coefficient μ=200. Algo. 5 first computes the new VNF placement p', shown in the green squared dots. While the VNFs migrate from their current switches towards the new ones in p', Algo. 5 finds all the VNF parallel migration frontiers m and records $C_b(p,m)$ on x-axis and $C_a(m)$ on y-axis. Fig. 6(b) shows that $C_a(m)$ decreases with the increase of $C_b(p,m)$, and $C_a(m)$ cannot be reduced without increasing $C_b(p,m)$ in each of the VNF parallel migration frontiers. Thus they are indeed Pareto fronts. However, instead of adopting any point on the Pareto front as the VNF migration scheme, Algo. 5 takes one step further and checks all the frontier points on the Pareto front and finds the one with the minimum cost.

Below we derive an optimal condition for Algo. 5 using scalarization [15]. Scalarization finds efficient solutions for a MOOP by linearly combining its multiple objective functions into a single objective function. We observe that Eq. 8 is indeed a scalarization of $C_a(m)$ and $C_b(p,m)$. It is well-known that if the solution function of a MOOP has a convex Pareto front, scalarization can identify optimal solutions [19]. We present the below theorem, which gives a sufficient condition for the optimality of Algo. 5, without proof.

Theorem 5: If the Pareto front of VNF migration generated by Algo. 5 is convex, Algo. 5 gives minimum total cost.

We demonstrate the effectiveness of Algo. 5 in Section VI by showing it outperforms two VM migration schemes [17], [24] and performs close to below exhaustive algorithm.

```
Algorithm 6: Exhaustive VNF Migration for TOM.
Input: A PPDC $G(V,E)$ with VM pair placement $s(v)$,
       $v \in \mathcal{V}$, and VNF placement $p(j)$, $1 \leq j \leq n$, $\mu$.
Output: A VNF migration $m$ and its total cost $C_t(m)$.
1. $C_t(p,m) = +\infty$;
2. Among all $|V_s| \cdot (|V_s|-1) \cdot ..., \cdot(|V_s|-n+1)$ VNF
      migrations, find $m$ with minimum cost $C_t(p,m)$.
```

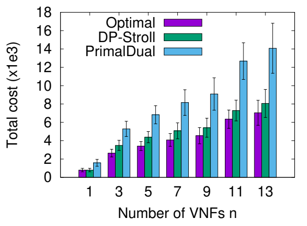


Fig. 7. Comparing TOP-1, l=1, k=8.

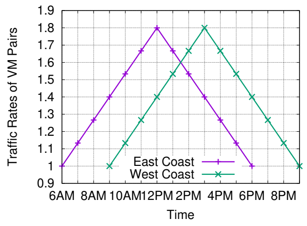


Fig. 8. Daily traffic rate pattern.

3. **RETURN** m and $C_t(p, m)$.

VI. Performance Evaluation

State-of-the-Arts. We first present the existing work of VNF placement and VNF migration that we compare with.

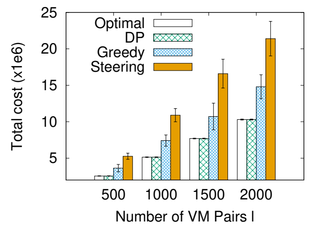
VNF Placement. Steering [55] considers that two services (i.e., VNFs) are dependent if they appear consecutively in a requested SFC, the degree of this dependency is the amount of traffic going through it. It picks the service with the highest dependency degree and finds its best location (i.e., minimizing the average time) until all services are placed. In our single-SFC model, Steering thus finds the best location for VNFs one by one. Liu [34] et al. proposes a two-step greedy algorithm. First, the MBs are sorted in descending order of their importance factor, which is the number of policies that use this MB. Second, it calculates each MB's *cost score* for each switch, and the switch with minimum cost score is selected to place the MB. The cost score is the increment of the total end-to-end delay by adding this MB plus the weighted average delay of all unplaced MBs to this MB.

TABLE II
Summary of Compared Algorithms.

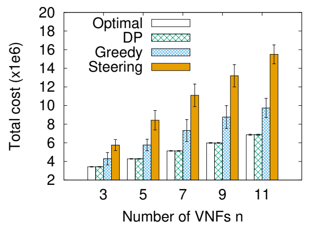
Problems	Our solutions	Existing work
TOP-1	DP-Stroll, Optimal	PrimalDual [10]
TOP	DP, Optimal	Steering [55], Greedy [34]
TOM	mPareto, Optimal	PLAN [17], MCF [24]

VNF Migration. Existing research proposes to use VNF migration in dynamic network traffic environment [20], [21], [35], [14]. However, they either does not consider migration cost [14] or have objectives different from ours [20], [21], [35], thus are not suitable for comparison. Instead, there are two recent research [17], [24] that migrate communicating VMs to reduce dynamic cloud traffic. PLAN [17] migrates VMs to hosts with available resources to maximize the *utility*, which is the reduction of the VM's communication cost minus its migration cost. Flores et al. [24] show that minimizing the total communication and migration cost of VMs is a minimum cost flow (MCF) problem [5]. As both works aim to reduce dynamic network traffic as our VNF migration approach does, we compare with them to see which is more effective.

Summary of Algorithms. Table II summarizes all algorithms. We refer to Algo. 4 and 6 as **Optimal**. For TOP-1, we refer to primal-dual-based Algo. 1 as **PrimalDual** and DP-based Algo. 2 as **DP-Stroll**, and compare DP-Stroll with the $2+\epsilon$ guarantee (i.e., two times of Optimal) of PrimalDual. For TOP,



(a) Varying l, $n = 7$.



(b) Varying n, $l = 1000$.

Fig. 9. Comparing VNF placement, $k = 8$.

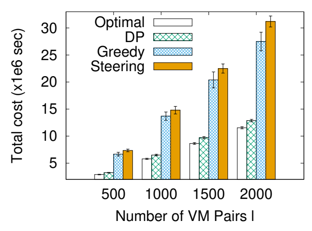
we refer to our DP-based Algo. 3 as **DP** and compare it with **Steering** [55] and **Greedy** [34]. For TOM, we refer to our Algo. 5 as **mPareto**, as it finds the minimum cost Pareto front, and compare it with **PLAN** [17] and **MCF** [24].

Experiment Setup. We study fat-tree PPDCs of k=8 with 128 hosts and k=16 with 1024 hosts. As 80% of cloud data center traffic originated by servers stays within the rack [8], we place 80% of the VM pairs into hosts under the same edge switches. Following diverse flow characteristics found in Facebook data centers [43], we assume the traffic rates of VM flows are in the range of [0, 10000], where 25% of VM flows have light traffic rates in [0, 3000), 70% medium traffic rates in [3000, 7000], and 5% heavy rates in (7000,10000]. Each data point in the plots is an average of 20 runs with a 95% confidence interval. We consider both unweighted (i.e., number of hops) and weighted (i.e., time delays on edges) PPDCs.

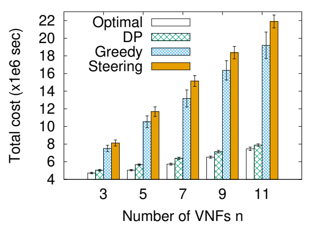
SFC Use Cases [3]. Real-world SFCs are broadly categorized into two types viz. access SFCs and application SFCs. As a typical SFC could have 5 to 6 access functions and 4 to 5 application functions, we consider up to 13 VNFs in an SFC.

Migration Coefficient μ. We quantify μ using the relative size of data or memory transferred in VM communication and VNF migration. As a typical data packet is around 1KB [8] and the size of transferred memory in migrating a containerized VNF is around 100MB [38], we set μ between 10^4 and 10^5.

Comparing TOP-1 Algorithms. Fig. 7 compares TOP-1 (i.e., n-stroll) algorithms in an unweighted PPDC with one VM pair. We observe that with the increase of the number of VNFs n, the communication cost of this VM pair increases as its traffic needs to traverse more VNFs. It also shows that DP-Stroll performs very close to Optimal, yielding only around 8% of more cost, and solidly outperforms the $2+\epsilon$ performance guarantee provided by PrimalDual.

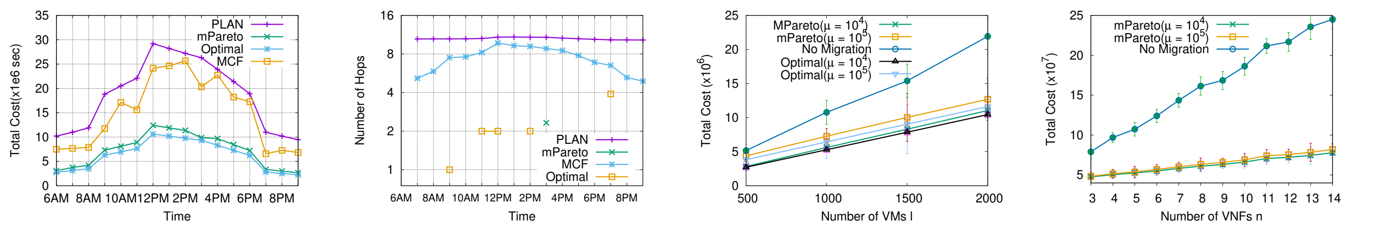


(a) Varying l, $n = 7$.



(b) Varying n, $l = 1000$.

Fig. 10. Comparing VNF placement with time delays, $k = 8$.



(a) Total cost comparison.

(b) # of VNF or VM migrations, y axis is in exponential scale with base 2.

(c) Compare with Optimal and NoMigration, $n = 7$.

(d) Compare with NoMigration, $l = 1000$.

Fig. 11. The effect of VNF migration in dynamic cloud traffic with $k = 16$ and $\mu = 10^4$ or 10^5.

Comparing TOP Algorithms. Fig. 9 compares the total VM communication costs of different VNF placement algorithms viz. Optimal, DP, Greedy, and Steering in unweighted PPDCs. Fig. 9 varies l and Fig. 9(b) varies n, respectively. In both cases, DP performs very close to Optimal, while both outperforming significantly Greedy and Steering. Fig. 10 considers weighted PPDCs by adopting the parameter setting in Greedy [34], wherein link delays follow a uniform distribution with a mean value of 1.5 ms and variance of 0.5 ms. It shows that the DP yields 6% to 12% more costs than that of the Optimal when varying number of VM flows l, and 12% to 16% more costs when varying number of VNFs n. DP yields costs 56% to 64% smaller than those of the Steering and Greedy, respectively.

Dynamic Traffic Model. We model dynamic cloud traffic as cycle-stationary with diurnal patterns [25]. Eramo et al. [20] proposed a sinusoidal model of 24 hours to quantify the scale factor of traffic rates τ_h at h^{th} hour. We instead consider $N = 12$ hours daily, where VM traffic rates increase gradually from 6 AM to noon and then decrease gradually from noon to 6 PM. Eqn. 9 shows our model, where τ_{min} is 0.2 as in [20]. Besides, to model the real effect of U.S. time zone difference upon cloud user activities, we assume that half of the VM flows (jobs submitted by users on the east coast) are three hours earlier than the other half (on the west coast). Fig. 8 visualizes the daily VM traffic rate pattern when changing their traffic rates following Eqn. 9.

$$\tau_h = \begin{cases} 0 & h = 0, \\ 2\frac{h}{N}(1 - \tau_{min}) & h = 1, ..., \frac{N}{2}, \\ 2\frac{N-h}{N}(1 - \tau_{min}) & h = \frac{N}{2} + 1, ..., N. \end{cases} \quad (9)$$

Effects of VNF Migrations on Dynamic Traffic. Fig. 11(a)-(d) investigate the effect of VNF migration on dynamic cloud traffic in a k=16 data center. Fig. 11(a) compares the total communication and migration costs of mPareto, PLAN, MCF, and Optimal. It shows that the mPareto performs within 5-10% of the Optimal, and both outperform PLAN and MCF by 52%-63%. Fig. 11(b) shows that the number of VNF migrations in mPareto is much smaller than the number of VM migrations in PLAN and MCF. Fig. 11(a) and (b) together show a smaller number of VNF migrations reduces more network traffic than a larger number of VM migrations, demonstrating that VNF migration is more effective in reducing dynamic traffic than VM migration. This is because migrating one VNF can reduce the total traffic of all the VM flows going through it while migrating one VM pair only affects the communication traffic of this pair.

Fig. 11(c) varies the number of VM pairs l and shows again that mPareto performs close to the Optimal. We also observe that total traffic costs for both mPareto and Optimal get slightly smaller when changing μ from 10^5 to 10^4, as a smaller migration coefficient encourages VNF migrations. Finally, we investigate how much traffic reduction VNF migration achieves by comparing the mPareto with the NoMigration. Fig. 11(c) and (d) vary l and n respectively, and it shows that the VNF migration reduces the total cost of VM flows by up to 73% compared to NoMigration. This demonstrates that VNF migration is indeed an effective technique to reduce dynamic cloud traffic in PPDC.

VII. Conclusions and Future Work

We proposed a new traffic-optimal VNF framework for dynamic PPDCs consisting of VNF placement and VNF migration. We formulated them as new graph-theoretical problems and proposed time-efficient approximation, Pareto-optimal, and DP-based algorithms. Using flow characteristics found in production data centers and realistic traffic patterns, we showed our algorithms outperform the state-of-the-art significantly and achieve network resource optimization for a PPDC's lifetime. As future work, we will consider a more general scenario wherein each switch can install multiple VNFs, and different VM flows can request different SFCs. We will investigate how VNF replication can alleviate dynamic VM traffic in PPDCs and study to which extent VNF replication could be beneficial in terms of dynamic traffic mitigation when compared to VNF migration. Finally, we will study how the traffic-reduction effect of VNFs (e.g., packet filtering) [36] can influence the problem formulations and solutions of VNF placement and migration in PPDCs.

Acknowledgment

This work was supported by NSF Grant CNS-1911191.

References

[1] Cisco global cloud index: Forecast and methodology, 2016 to 2021 white paper. https://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/white-paper-listing.html.

[2] Hosting large meetings in zoom. https://support.zoom.us/hc/en-us/articles/201362823-Hosting-large-meetings.

[3] Service function chaining use cases in data centers (ietf). https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06section-3.3.1.

[4] Zoom cloud meetings. https://zoom.us/.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[6] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, 2008.

[7] M. Bateni and J. Chuzhoy. Approximation algorithms for the directed k-tour and k-stroll problems. In *Proc. of APPROX/RANDOM 2010*.

[8] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of ACM IMC 2010*.

[9] F. Carpio, A. Jukan, and R. Pries. Balancing the migration of virtual network functions with replications in data centers. In *Proc. of IEEE/IFIP NOMS*, 2018.

[10] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proc. of IEEE FOCS 2003*.

[11] D. Chemodanov, P. Calyam, and F. Esposito. A near optimal reliable composition approach for geo-distributed latency-sensitive service chains. In *IEEE INFOCOM 2019*.

[12] D. Chemodanov, P. Calyam, and F. Esposito. A near optimal reliable composition approach for geo-distributed latency-sensitive service chains. In *IEEE INFOCOM 2019*.

[13] Y. Chen, J. Wu, and B. Ji. Virtual network function deployment in tree-structured networks. In *Proc. of ICNP 2018*.

[14] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry. Real-time virtual network function (vnf) migration toward low network latency in cloud environments. In *Proc. of IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017.

[15] J. Cho, Y. Wang, I. Chen, K. S. Chan, and A. Swami. A survey on modeling and optimizing multi-objective systems. *IEEE Communications Surveys Tutorials*, 19(3):1867–1901, 2017.

[16] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz. Near optimal placement of virtual network functions. In *Proc. of INFOCOM 2015*.

[17] L. Cui, F. P. Tso, D. P. Pezaros, W. Jia, and W. Zhao. Plan: Joint policy- and network-aware vm management for cloud data centers. *IEEE Transactions on Parallel and Dis. Sys.*, 28(4):1163–1175, 2017.

[18] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros. Dynamic, latency-optimal vnf placement at the network edge. In *IEEE INFOCOM 2018*.

[19] M. Ehrgott. *Multicriteria Optimization*. Springer, 2005.

[20] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca. An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Transactions on Networking*, 25(4):2008–2025, 2017.

[21] B. Farkiani, B. Bakhshi, S. Ali MirHassani, T. Wauters, B. Volckaert, and F. De Turck. Prioritized deployment of dynamic service function chains. *IEEE/ACM Transactions on Networking*, pages 1–15, 2021.

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

[23] H. Feng, J. L., A. M. Tulino, D. Raz, and A. F. Molisch. Approximation algorithms for the nfv service distribution problem. In *Proc. of IEEE INFOCOM 2017*.

[24] H. Flores, V. Tran, and B. Tang. Pam & pal: Policy-aware virtual machine migration and placement in dynamic cloud data centers. In *Proc. of IEEE INFOCOM 2020*.

[25] S. Gebert, R. Pries, D. Schlosser, and K. Heck. Internet access traffic measurement and analysis. In *2012 Int. Conf. Traffic Monitor. Anal.*

[26] C. Gonzalez and B. Tang. Aggvnf: Aggregate virtual network function allocation and migration in cloud data centers. Submitted to Infocom 2022.

[27] A. Gushchin, A. Walid, and A. Tang. Scalable routing in sdn-enabled networks with consolidated middleboxes. In *Proc. of ACM Hotmiddlebox*, 2015.

[28] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi. Traffic steering for service function chaining. *IEEE Communications Surveys Tutorials*, 21(1):487–507, 2019.

[29] M. Huang, W. Liang, Y. Ma, and S. Guo. Maximizing throughput of delay-sensitive nfv-enabled request admissions via virtualized network function placement. *IEEE Transactions on Cloud Computing*, 2019.

[30] N. Huin, B. Jaumard, and F. Giroire. Optimal network service chain provisioning. *IEEE/ACM Trans. on Netw.*, 26(3):1320–1333, June 2018.

[31] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat. Experience with a globally-deployed software defined wan. In *Proc. of ACM SIGCOMM 2013*.

[32] B. Jaumard and H. Pouya. Migration plan with minimum overall migration time or cost. *J. Opt. Commun. Netw.*, 10:1 – 13, 2018.

[33] T. Kuo, B. Liou, K. C. Lin, and M. Tsai. Deploying chains of virtual network functions: On the relation between link and server usage. *IEEE/ACM Transactions on Networking*, 26(4):1562–1576, Aug 2018.

[34] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin. Improve service chaining performance with optimized middlebox placement. *IEEE Transactions on Services Computing*, 10(4):560–573, 2017.

[35] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu. On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management*, 14(3):543–553, 2017.

[36] W. Ma, J. Beltran, D. Pan, and N. Pissinou. Traffic aware placement of interdependent nfv middleboxes. *IEEE Transactions on Network and Service Management*, 16(4):1303–1317, Dec. 2019.

[37] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer. Remedy: Network-aware steady state vm management for data centers. In *Proc. of the NETWORKING 2012*.

[38] R. J. Martins, C. B. Both, J. A. Wickboldt, and L. Z. Granville. Virtual network functions migration cost: from identification to prediction. *Computer Networks*, 181(9), 2020.

[39] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM 2010*.

[40] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Sur. and Tut.*, 18(1), 2015.

[41] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz. A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks. *IEEE Transactions on Network and Service Management*, 14(3):554–568, 2017.

[42] J. Reese. Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3):125–142, 2006.

[43] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *Proc. of SIGCOMM 2015*.

[44] G. Sallam and B. Ji. Joint placement and allocation of vnf nodes with budget and capacity constraints. *IEEE/ACM Transactions on Networking*, 2021.

[45] G. Sallam, Z. Zheng, and B. Ji. Placement and allocation of virtual network functions: Multi-dimensional case. In *Proc. of ICNP 2019*.

[46] 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 *Proc. of INFOCOM 2017*.

[47] L. Tang, X. He, P. Zhao, G. Zhao, Y. Zhou, and Q. Chen. Virtual network function migration based on dynamic resource requirements prediction. *IEEE Access*, 7:112348–112362, 2019.

[48] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes. Provably efficient algorithms for placement of service function chains with ordering constraints. In *IEEE INFOCOM 2018*, pages 774–782, 2018.

[49] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu. Delay-aware virtual network function placement and routing in edge clouds. *IEEE Transactions on Mobile Computing*, 2019.

[50] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu. Delay-aware virtual network function placement and routing in edge clouds. *IEEE Transactions on Mobile Computing*, 20(2):445–459, 2021.

[51] B. Yi, X. Wang, M. Huang, and A. Dong. A multi-criteria decision approach for minimizing the influence of vnf migration. *Computer Networks*, 159:51 – 62, 2019.

[52] B. Yi, X. Wang, M. Huang, and K. Li. Design and implementation of network-aware vnf migration mechanism. *IEEE Access*, 8:44346–44358, 2020.

[53] Q. Zhang, F. Liu, and C. Zeng. Adaptive interference-aware vnf placement for service-customized 5g network slices. In *IEEE INFOCOM 2019*, 2019.

[54] X. Zhang, Z. Xu, L. Fan, S. Yu, and Y. Qu. Near-optimal energy-efficient algorithm for virtual network function placement. *IEEE Transactions on Cloud Computing*, pages 1–1, 2019.

[55] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patney, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula. Steering: A software-defined networking for inline service chaining. In *Proc. of IEEE ICNP 2013*.

[56] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao. Towards latency optimization in hybrid service function chain composition and embedding. In *IEEE INFOCOM 2020*.