Low-Latency Service Function Chain Migration in Edge-Core Networks based on Open Jackson Networks

Wenjie Liang[a], Lin Cui[a,*], Fung Po Tso[b]

[a]*Department of Computer Science, Jinan University, Guangzhou, China*
[b]*Department of Computer Science, Loughborough University, UK*

Abstract

Multi-access Edge Computing (MEC) offers cloud computing capabilities at the edge of the network. Growing demand for low-latency services requires Service Function Chains (SFCs) to be scaled up beyond MEC network to core network. To adapt to network dynamics and provide low-latency services, being able to migrate SFCs when needed is of paramount importance. However, migration of SFCs among edge and core networks such that average latency is optimized as well as considering resource consumption is an intractable challenge because improper migration of Virtual Network Functions (VNFs) results in failure of meeting the requirements of network policies. In this paper, we investigate SFCs in edge-core networks and model the *Latency-aware Edge-Core SFCs Migration* problem based on open Jackson networks. Two SFC migration algorithms, i.e., Profit-driven Heuristic Search (PHS) and Average Utilization Based (AUB), are proposed to efficiently optimize average latency of all SFCs in edge-core networks. Extensive evaluation results show that PHS optimizes average latency by 19.5%, while AUB can further reduce average latency by up to 36.9% by allowing a marginally higher number of VNF migrations.

Keywords: SFC Migration, MEC, Core Network, Latency

1. Introduction

Recent years have seen proliferating demand from mobile users for latency-sensitive applications and services, such as augmented reality (AR), autonomous vehicles (AVs) and ultra-low latency video streaming [1], etc. In response to this demand, Multi-access Edge Computing (MEC) architecture has emerged to provide network services in closer proximity to user equipment (UE). Meanwhile, network services are usually formed by ordered collections of network functions (NFs) like firewalls, proxy servers and load balancers, etc., namely Service Function Chains (SFCs). With the advent of Network Function Virtualization (NFV) [2], most NFs can be virtualized as Virtual Network Functions (VNFs).

When more and more SFCs are deployed, scaling up SFCs in MEC-enabled networks is difficult as edge networks usually fall short of computing resources [3]. In comparison, core network is rich in computing resources, allowing VNFs to be offloaded from edge network [4]. Hence, combining edge with core network forms an edge-core network, which provides greater flexibility for latency optimization and resource utilization of SFCs. A promising SFC use case in edge-core networks is the delivery of Value-Added Services (VAS), such as HTTP header enrichment in edge network to identify and charge subscribers, and Carrier Grade NAT (CG-NAT) and NAT64 in core network for technical reasons (e.g., traffic flows traverse different networks supporting distinct address families like IPv4/IPv6) [5]. However, provisioning of low-latency services in edge-core networks is prone to network congestion and compute resource contention caused by network dynamics. Overcoming these issues requires SFC scheduling mechanisms that can proactively adapt to network conditions.

With regard to scheduling of SFCs and reducing latency of network services in edge and core networks, some works focus on SFC deployment to optimize network latency [6], deployment cost [7], and energy consumption [8, 9]. These works employ static placement of SFCs based on the assumption that network demand is known and resources are not oversubscribed. This intrinsic shortcoming prevents them from dealing with dynamic network conditions efficiently due to their inability to react to changing network demand and resource hotspots. VNF replication can mitigate the problem with good reliability and scalability [10, 11], but it is challenging to maintain a large number of internal states across multiple servers and ensure consistency when replicating stateful VNFs. In comparison, VNF migration [12] is an effective way to deal with dynamics because VNFs can be flexibly migrated to optimal servers while reserving application states.

However, SFC migration in edge-core networks remains unexplored and is challenging. There are works [4, 13] studying migration of individual VNFs across edge and core networks, but they do not consider the constraint of the orders of VNFs in an SFC, which may result in violation of the constraint and increase of end-to-end latency for users. SFC migration in either edge network or cloud data center has been investigated [14, 15, 16]. However, these works cannot be directly applied to edge-core

[*]Corresponding author
Email address: `tcuilin@jnu.edu.cn` (Lin Cui)

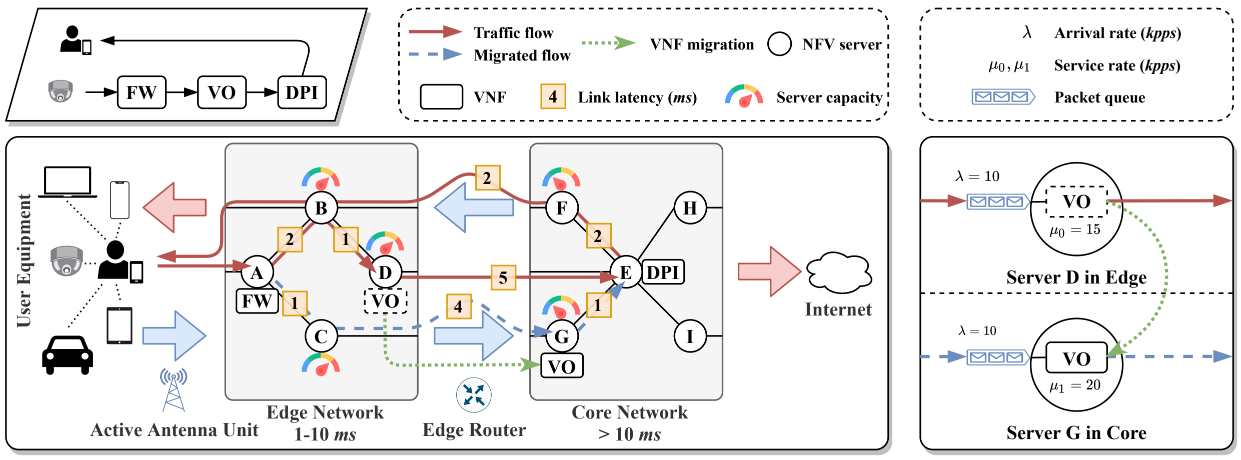

Figure 1: A typical example of user requests of live video streaming service served by an SFC, i.e., firewall (FW), video optimizer (VO) and deep packet inspector (DPI) sequentially: (i) the left part shows link latency is reduced from 12 *ms* (marked by the red arrows) to 10 *ms* (marked by the blue dashed arrows) after VNF migration; (ii) the right part indicates VNF processing latency is optimized as VO is allocated more resources in core network than edge network.

networks due to two major challenges in edge-core networks from the communication perspective [17]. (i) Compared to either edge network or cloud data center, traffic patterns are more complex in edge-core networks, e.g., flows can be transmitted between UEs and the Internet, entering or leaving the edge and core networks arbitrarily. (ii) Due to different underlying network architectures in edge and core networks, the latency from UEs to different parts of the edge-core networks varies greatly (as shown in Fig. 1). These challenges increase the complexity and difficulty for modeling flow latency and making optimal decisions on SFC migration across edge-core networks. Fig. 1 demonstrates an example of users requesting live video streaming service served by an SFC to illustrate the importance of SFC migration in edge-core networks. Migrating VNFs to appropriate NFV servers across edge-core networks can optimize link latency and processing latency. User-generated video streams are steered through an SFC consisting of: firewall (FW)→video optimizer (VO)→deep packet inspector (DPI), and forwarded to other users who subscribe live video services (marked by the red arrows). Initial placement of VNFs is not optimal, leading to longer latency. After the video optimizer is migrated from edge network to core network, the SFC end-to-end latency is reduced, including both link latency (the migrated flows marked by the blue dashed arrows) and processing latency (processing rate of video optimizer improves as more computing resources are allocated).

In this paper, we investigate latency-aware SFCs migration in edge-core networks. To address the above challenges, we first model SFCs using open Jackson networks [18]. The open Jackson network well captures network characteristics in the aspect of flow patterns in SFCs. Besides, the average latency of SFCs in both edge and core networks is reduced by SFC migration without violating service-level agreements (SLAs) compliance. Two heuristic algorithms are proposed to solve the SFC migration problem efficiently. To the best of our knowledge, this is the first paper that comprehensively considers SFC migration based on the interplay of edge with core network. The main contributions of this paper are summarized as follows:

- We apply open Jackson networks to model SFCs in edge-core networks and formulate the *Latency-aware Edge-Core SFC Migration* problem. The open Jackson network is used to analyze latency of SFCs to minimize the waiting time in VNF queues.
- Two novel algorithms, i.e., Profit-driven Heuristic Search (PHS) and Average Utilization Based (AUB) for SFC migration, are proposed to optimize average latency of all deployed SFCs.
- Extensive evaluations are conducted to demonstrate effectiveness of both algorithms: PHS can migrate a small number of VNFs (i.e., 6.7% migration rate) to optimize more latency by nearly 19.5%. By allowing a bit more VNF migrations, AUB can reduce average latency of SFCs by up to 36.9% compared to the comparative method.

The remainder of this paper is structured as follows. In Section 2, an overview of related works is given. In Section 3, we formulate SFC migration problem and define an objective function. Then, an analysis of minimizing average latency by migration of VNFs will be given in detail. In Section 4, two effective algorithms PHS and AUB are proposed in detail. Section 5 presents our evaluation environment and the performance of the proposed algorithms. In Section 6, a brief discussion on current study and future works is provided, and then Section 7 concludes the paper.

2. Related Works

2.1. SFC Deployment

Many works focus on SFC deployment for the purposes of resource, energy, and latency optimization. Li et al. [19] studied placement of VNFs and resource optimization in edge computing enabled networks, and proposed a polynomial time heuristic solution to solve the problem. By leveraging deep learning and Software-Defined Networking (SDN), Zhang et al. [8] proposed an intelligent architecture to solve the multi-domain SFC deployment problem. Zeng et al. [9] designed a VNF placement and routing algorithm based on a genetic algorithm and simplex method to optimize energy consumption and Quality of Service (QoS). The work [6] emphasizes latency-aware and reliable SFC placement against VNF failures. They aimed to minimize total deployment cost while mitigating high computational complexity of the Integer Linear Programming (ILP) problem. Cui et al. [20] studied NF placement in heterogeneous network environments with an objective to minimize network cost. Zhang et al. [21] modeled VNF chains in data center networks by applying open Jackson networks and formulated the VNF placement problem as bin-packing problem to improve resource utilization and reduce latency. Harutyunyan et al. [22] aimed to minimize the service provisioning cost and satisfy users' data rate requirements through placement and scaling of VNFs in MEC and 5G core network.

2.2. *SFC Migration*

VNF migration, such as cold/live migration and virtual machine/container migration, has been widely studied [13, 23]. Live migration of stateful VNFs, which consists of pre-copy, post-copy, and hybrid schemes, etc., is the most common approach used in real world environments. Sarrigiannis et al. [4] presented an MEC-enabled 5G architecture that efficiently handles the placement and migration of network and application VNFs across edge and core tiers. These VNFs are orchestrated by an NFV Orchestrator (NFVO) with admission management functionalities that manage NFs and resources on-the-fly. Zhang et al. [13] applied Para-Virtualization QuickAssist Technology (PV-QAT) to accelerate the live migration of VNFs which are deployed in virtual machines.

There are some works studying SFC migration. Addad et al. [14] introduced and evaluated diverse SFC migration patterns regarding user mobility, server resources and application requirements. They aimed to achieve an objective of 1 *ms* latency for the 5G network. Hawilo et al. [24] studied VNF migration and re-instantiation to achieve minimal downtime and minimize SFC delays. Chen et al. [25] proposed a deep reinforcement learning framework called MSDF to address multiple SFC migrations. Their work outperforms typical heuristic algorithms in the aspect of reducing network operation cost and balancing QoS of users. To study user mobility, Zhao et al. [15] researched SFCs remapping in cloud-fog computing environments by using the minimum number of VNFs migration strategy and the two-step migration strategy. They focused on minimization of reconfiguration time and downtime. Similarly, Chen and Liao [16] tackled the user mobility problem using SFC placement and migration to maximize user satisfaction. Chemodanov et al. [26] studied SFC composition and maintenance in geo-distributed edge/core cloud infrastructures, for the purposes of provisioning of reliable latency-sensitive SFCs. They presented a metapath-based SFC maintenance algorithm by allowing migration of part of the service chain to maintain its services. Guo et al. [27] designed a dynamic hierarchical SFC orchestration algorithm based on deep reinforcement learning to minimize network cost and improve QoS in the edge cloud and core cloud.

Table 1: Notations

Network parameters	Description
$\mathbb{G} = (\mathbb{V}, \mathbb{E})$	Network as a graph.
$\mathbb{V} = \{v_1, v_2, v_i \ldots, v_{\lvert\mathbb{V}\rvert}\}$	All NFV servers in the network.
$\mathbb{E} = \{e_1, e_2, e_j \ldots, e_{\lvert\mathbb{E}\rvert}\}$	A set of paths between any two servers.
C_i	Capacity as $\{CPU, memory, I/O\}$ of server $v_i \in \mathbb{V}$.
Y_i	A list of VNFs residing on server $v_i \in \mathbb{V}$.
W_j	Link latency of path $e_j \in \mathbb{E}$.
B_j	Bandwidth of path $e_j \in \mathbb{E}$.
VNF parameters	**Description**
$\mathbb{F} = \{f_1, f_2, f_m, \ldots, f_{\lvert\mathbb{F}\rvert}\}$	A set of VNFs.
R_m	Resources as $\{CPU, memory, I/O\}$ required for instantiation of VNF $f_m \in \mathbb{F}$.
A_m	Actual resources allocated to VNF $f_m \in \mathbb{F}$. It is required that $A_m \geq R_m$ for f_m to operate normally.
SFC parameters	**Description**
$\mathbb{S} = \{s_1, s_2, s_k, \ldots, s_{\lvert\mathbb{S}\rvert}\}$	A set of SFCs, each of which contains a series of ordered VNFs.
P_k	A path of SFC $s_k \in \mathbb{S}$.
ω_k	Maximum allowed end-to-end latency for SFC $s_k \in \mathbb{S}$.
ξ_k	Flow rate of traffic flows served by SFC $s_k \in \mathbb{S}$.
Others	**Description**
λ_{f_m}, μ_{f_m}	Average arrival rate and service rate related to VNF f_m.
T_{f_m}	Average response latency of each packet in VNF f_m.
$b^m_{i,j}$	Binary variable that equals 1 if f_m is migrated from server v_i to v_j.
$\mathbb{H}_m$	A subset of NFV servers to which VNF f_m can be migrated.

Different from above works, this paper focuses on providing low-latency SFCs in edge-core networks through migration of VNFs by modeling SFCs with open Jackson networks.

3. Problem Model

In this section, deployment of SFCs in edge-core networks is modeled using open Jackson networks. And we formulate the *Latency-aware Edge-Core SFC Migration* problem with an objective to minimize average latency of all SFCs. Table 1 lists key notations that will be used in the following of the paper.

3.1. *Edge-Core Networks and SFCs*

MEC provides an ultra-low latency and high bandwidth compute environment for service providers and customers. But computational functionalities of MEC servers (e.g., deployed at base stations and aggregation points [28]) are limited. Meanwhile, core network is considered to have abundant computing resources through on-demand upscaling [4] while retaining low latency compared to cloud data centers. Edge-core networks combine the advantages of the two networks, as shown in Fig. 1. However, traffic patterns for edge and core networks are different. Enforced by SFCs, flows generated by UEs are first steered to the radio access network (RAN), where MEC servers are deployed. Some flows may be forwarded to core network (or backbone), which is connected to RAN via backhaul links and edge routers [29]. Based on policy requirements of services, these traffic flows can be routed to the Internet or back to UEs. Therefore, edge network not only needs to handle traffic flows from UEs, but also to relay flows from core network to UEs. These intrinsic characteristics of traffic patterns influence the decisions of SFC scheduling between edge and core networks.

The edge-core networks are defined as a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \mathbb{V}_M \cup \mathbb{V}_C$ refers to a set of NFV servers either in edge or core network, and $\mathbb{E} = \mathbb{E}_M \cup \mathbb{E}_C$ is a set of paths among NFV servers. For each NFV server $v_i \in \mathbb{V}$, its computing resource[1] is denoted as capacity C_i. The latency for each path $e_j \in \mathbb{E}$ is denoted by W_j, including both transmission delay and propagation delay. The latency within either edge or core network is usually low, as shown in Fig. 1, however, latency from UEs to core network is higher than that from UEs to edge network due to longer distance. Latency between two NFV servers in edge network and core network can be measured by network monitoring tools like sFlow[2] and MTR[3]. Bandwidth of e_j is denoted by B_j.

An NFV server can spawn several VNF instances by spinning up one or more virtual machines (VMs) or containers. Y_i denotes a list of VNFs that are already instantiated in server v_i. For each VNF instance $f_m \in \mathbb{F}$, the amount of resources required for instantiation is denoted by R_m. When traffic flows arrive at server v_i, they are served by corresponding VNFs immediately. Average arrival rate of traffic flows to VNF f_m (packet per second, *pps*) is denoted by λ_{f_m}. Let μ_{f_m} be the average service rate or throughput of f_m. Usually, service rates of different VNFs are non-deterministic, and they may vary to cater for various requirements defined in SLAs. The service rate of each VNF is determined by the amount of assigned computing resources, i.e., VNFs with higher service rate are likely to consume more computing resources [31]. A_m denotes actual resources allocated to VNF f_m. Without loss of generality, computing resources of a server are assumed to be evenly allocated among all VNF instances that reside on the server.

To understand the relationship between service rate and allocated resources, we conduct an experiment[4] with three VNFs,

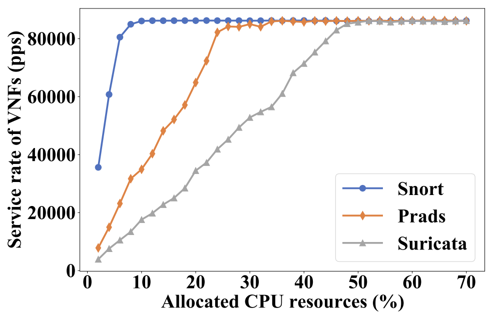

Figure 2: Impact of allocated CPU resources on packet processing rates of different types of VNFs.

i.e., Snort[5], Prads[6] and Suricata[7]. Results in Fig. 2 show that allocated CPU resources have different impacts on service rates of the three VNFs. The differences can be summarized as Equation (1), where parameters k, b, c and d may vary with the types of VNFs. k reflects the sensitivity of service rate of a VNF to its allocated resources. d is the maximum effective resources allocated to the VNF and c is the upper bound of service rate.

$$\mu_{f_m} = \begin{cases} kA_m + b, & A_m < d. \\ c, & A_m \geq d. \end{cases} \tag{1}$$

The set of SFCs is defined as $\mathbb{S}$. For each SFC $s_k \in \mathbb{S}$, it consists of a sequence of connected VNFs $\mathbb{F}_k \subseteq \mathbb{F}$, which performs an ordered action on the data stream. According to IETF RFC 7665 [32], ingress and egress of SFC s_k act as SFC boundary nodes to handle traffic entering and leaving the SFC-enabled domain respectively. The ingresses and the egresses can be access points (APs) or switches [33]. When traffic flows are steered through an SFC, they follow a specific path from ingress to egress of the SFC. P_k denotes an SFC path of s_k that contains an ordered list of paths among NFV servers. Besides, each SFC s_k has maximum allowed end-to-end latency ω_k and is programmed to serve up to a certain number of traffic flows with flow rate ξ_k. The probability that flows are steered from server v_i to v_j is denoted by p_{ij}. In practice, a VNF can be shared by more than one SFCs, but migration of the shared VNFs may interfere with the schedule decisions. Since this paper focuses on scheduling under edge-core networks, for simplicity, we assume that an independent VNF instance can only be assigned to each SFC.

3.2. *Modeling SFCs as Open Jackson Networks*

In queuing networks, an open network describes how customers arrive from an external source are served and then leave the system. An open Jackson network [18] is a special class of queuing networks considering several queues with one class of customers.

We model VNFs as M/M/1 (Kendall's notation) queues. According to the *Burke's Theorem* [34], the departure process of

[1]Server resources are measured in units [30]. For example, one unit of resource can be defined as 1 CPU core and 512 MB RAM.

[2]https://sflow.org

[3]https://www.bitwizard.nl/mtr/

[4]This experiment is conducted on a server with 4-core Intel Core i7-6700 CPU and 8GB memory. Each VNF is deployed in different *namespaces*, and *cgroups* are used to control the amount of resources for each VNF.

[5]https://www.snort.org/

[6]https://github.com/gamelinux/prads

[7]https://suricata-ids.org/

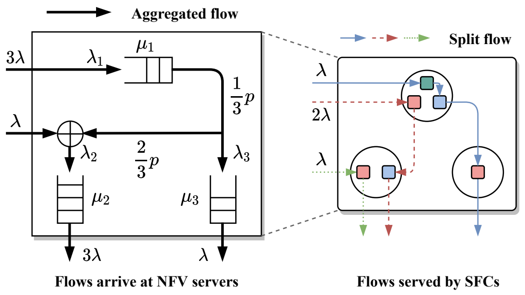

Figure 3: Aggregated traffic flows arrive to different NFV servers under open Jackson network: aggregated flows from the left part are corresponding to multiple split flows from the right part.

traffic flows from the network will be Poisson if the system is under equilibrium conditions. For M/M/1 type queues, it is assumed that: (i) any external traffic flows from outside entering the edge-core networks form a Poisson process [35] with an arrival rate $\alpha > 0$, service times at each VNF are exponentially distributed; (ii) queuing discipline of each packet in VNF queues follows a first-come, first-served (FCFS) rule; (iii) upon service completion at server v_i, a flow will either go to another server v_j with probability p_{ij} or leave the edge-core networks with probability $p_{i0} = 1 - \sum_{j=1}^{|\mathbb{V}|} p_{ij}$; (iv) the utilization of all VNFs is less than one, as shown in Equation (2).

$$\rho_{f_m} = \frac{\lambda_{f_m}}{\mu_{f_m}} < 1, \forall f_m \in s_k, \forall s_k \in \mathbb{S} \tag{2}$$

Each external traffic flow from outside of the edge-core networks is independently routed to VNF $f_m \in Y_i$ with probability p_{0i}. The overall arrival rate to VNF f_m is defined as λ_{f_m} in Equation (3), including both external arrivals (UEs to edge network, between edge and core network) and internal transitions (inside edge or core network). By solving these flow balance equations (i.e., Equation (3)), we can obtain the effective arrival rate to each VNF.

$$\lambda_{f_m} = \alpha p_{0i} + \sum_{j=1}^{|\mathbb{V}|} \lambda_{f_n} p_{ji}, \forall f_m \in Y_i, \forall f_n \in Y_j, \forall v_i \in \mathbb{V} \tag{3}$$

In edge-core networks, network packets likewise act as customers to be served by a sequence of VNFs (i.e., SFC). The chaining characteristic of an SFC implies that the arrival process of flows at a VNF is related to the service process at the former VNF, which is well captured by open Jackson network [21]. Fig. 3 shows the process of applying open Jackson networks to model SFCs. The left part illustrates an overview of traffic flows entering the queues under the open Jackson network. The right part shows NFV servers from the left part in more detail: the aggregated traffic flows are split and steered into corresponding VNFs of SFCs.

Under open Jackson networks, the whole network is at equilibrium, i.e., service rate μ_{f_m} is greater than arrival rate λ_{f_m} for all VNFs. Otherwise, the system will become unstable and severe packet loss happens because the buffer at each queue overflows as time goes on. With this precondition, average response latency of each packet in the queue of VNF f_m is therefore given by:

$$T_{f_m} = \frac{1}{\mu_{f_m} - \lambda_{f_m}} \tag{4}$$

3.3. Problem Formulation

To obtain the mean sojourn time spent by each packet in the system, we first count the average number of packets in edge-core networks. On the one hand, the average number of packets η_m in the VNF f_m can be calculated using Equation (5). Then, the total average number of packets in all servers can be obtained for both networks. On the other hand, the average number of packets in path e_j can be calculated by arrival rate and latency of e_j. Since all VNFs are at equilibrium, the average flow rate leaving the VNF will be equal to the average flow rate entering the VNF. For each SFC s_k, arrival rates of all traffic flows served by s_k are aggregated into λ_{s_k}. The latency of e_j varies from edge to core network. We break down the differences in the latency W_j for each path e_j, i.e., $P_k \cap \mathbb{E}_M$ for edge network and $P_k \cap \mathbb{E}_C$ for core network. The average number of packets L_M in edge network and L_C in core network can be formulated by Equation (6) and Equation (7), respectively.

$$\eta_m = \frac{\lambda_{f_m}}{\mu_{f_m} - \lambda_{f_m}} \tag{5}$$

$$L_M = \sum_{v_i \in \mathbb{V}_M} \sum_{f_m \in Y_i} \eta_m + \sum_{s_k \in \mathbb{S}} \sum_{e_j \in P_k \cap \mathbb{E}_M} \lambda_{s_k} W_j \tag{6}$$

$$L_C = \sum_{v_i \in \mathbb{V}_C} \sum_{f_m \in Y_i} \eta_m + \sum_{s_k \in \mathbb{S}} \sum_{e_j \in P_k \cap \mathbb{E}_C} \lambda_{s_k} W_j \tag{7}$$

As mentioned earlier, both edge and core networks render great difference with respect to computational capabilities and traffic directions. For a fine-grained analysis on SFC migration in the context of edge-core networks, all deployed SFCs are modeled using the open Jackson network separately. As shown in Fig. 1, the total arrival rate of traffic flows from UEs to edge network is denoted by Λ_U. The total arrival rate from edge to core network is denoted by Λ_M. Flows in core network will be either sent to the Internet or sent back to the UEs through edge network. Λ_C is the total arrival rate of traffic from core to edge network. By applying the *Little's law* [36] for open Jackson networks, we can obtain the average time spent by each packet from arrival until fully processed by SFCs in edge and core networks, as shown in Equation (8) and Equation (9), respectively.

$$T_M = \frac{L_M}{\Lambda_U + \Lambda_C} \tag{8}$$

$$T_C = \frac{L_C}{\Lambda_M} \tag{9}$$

Hence, the average latency Γ incurred by all deployed SFCs in edge-core networks is:

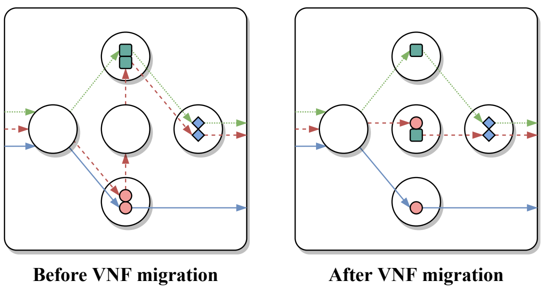

Figure 4: An example of SFC migration: three traffic flows are served by three SFCs separately and two VNFs of the SFC (marked by the red dashed arrows) are migrated to an idle server.

$$\Gamma = T_M + T_C \tag{10}$$

Suppose an SFC, say s_k, is successfully migrated, a new SFC path needs to be reconstructed to ensure that SLAs of both latency and bandwidth are satisfied. There are necessary factors that affect path reconstruction: (i) the shortest path between any two consecutive VNFs; (ii) fixed order of VNFs; (iii) available bandwidth of the path. Fig. 4 illustrates that two VNFs of the SFC are migrated to an idle server, for the purposes of path optimization and improving resource utilization. The latency reduction due to the migration is denoted by δ in Equation (11), where Γ and Γ' refer to average latency of all SFCs before and after migration, respectively.

$$\delta = \Gamma - \Gamma' \tag{11}$$

The goal is to maximize latency reduction through migration, i.e., maximizing δ as shown in Equation (12a).

$$\begin{aligned}
\text{maximize} \quad & \delta && \text{(12a)}\\
\text{subject to} \quad & \delta > 0 && \text{(12b)}\\
& \sum_{e_j \in P_k} W_j + \sum_{f_m \in s_k} T_{f_m} \le \omega_k, \forall s_k \in \mathbb{S} && \text{(12c)}\\
& \sum_{f_m \in Y_i} A_m \le C_i, \forall v_i \in \mathbb{V} && \text{(12d)}\\
& \sum_{s_k \in \mathbb{S}, e_j \in P_k} \xi_k \le B_j, \forall e_j \in \mathbb{E} && \text{(12e)}\\
& \sum_{f_m \in s_k} b^m_{i,j} \ge 1, \exists s_k \in \mathbb{S}, \forall v_i \in \mathbb{V}, \forall v_j \in \mathbb{V} && \text{(12f)}\\
& v_i \in \mathbb{H}_m, \forall f_m \in Y_i, \forall v_i \in \mathbb{V} && \text{(12g)}
\end{aligned}$$

In the above equations there are six constraints (12b)–(12g). Constraint (12b) indicates that average latency of all SFCs should be reduced after SFC migration. Network dynamics is one of the major factors that affects the performance of SFCs. Constraint (12b) does not guarantee whether the latency of a single SFC will be reduced or not. Services provided by all SFCs must meet the criteria of the predefined SLAs, i.e., not exceeding the maximum allowed end-to-end latency. Based on link latency W_j and response latency mentioned in Equation (4),

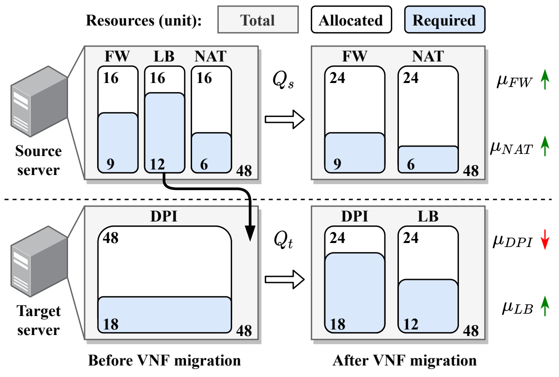

Figure 5: An example of VNF migration: allocated resources for FW, network address translator (NAT) and LB increase while DPI is allocated less resources after migration, which may result in changes to their service rates.

Constraint (12c) ensures that the latency of an SFC s_k is restricted to the maximum allowed end-to-end latency ω_k due to SLAs. Constraint (12d) is the capacity limitation of each NFV server. Constraint (12e) refers to that the bandwidth of all paths for SFCs cannot be exceeded. ξ_k denotes flow rate of traffic flows served by s_k. Constraint (12f) guarantees that at least one VNFs must be migrated, where $b^m_{i,j}$ is a conditional expression that represents whether f_m is migrated from server v_i to v_j or not.

$$b^m_{i,j} = \begin{cases} 1, & \text{if } f_m \text{ is migrated from } v_i \text{ to } v_j. \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

For VNFs to be migrated, some of them only work in a specific area to provide services because of their functionalities. For example, bastion hosts and intrusion prevention systems (IPS) should be placed at the entrance of core network to secure the networks. Constraint (12g) indicates that VNF f_m is allowed to be migrated to some specific servers. $\mathbb{H}_m$ is a subset of NFV servers from $\mathbb{V}$ to which f_m can be migrated.

The problem defined in (12a) can be easily proven to be NP-hard through reducing the multiple knapsack problem (MKP), whose decision version has already been proven to be strongly NP-complete.

There is a prerequisite that the network capacity is large enough for accommodating all SFCs, as SFCs are already scheduled in policies. Nevertheless, improper migration may lead to rising cost (e.g., migration overhead [37]) and failure of satisfying partial stringent requirements of policies [38].

3.4. Discussions

3.4.1. Profit of migrating a VNF

If a VNF $f_m \in s_k$ is migrated from source server v_s to target server v_t, there are four obvious effects: (i) f_m releases resources from v_s; (ii) v_t needs to allocate resources to f_m; (iii) link latency of the SFC path P_k may change; (iv) other SFCs that traverse v_s and v_t may be affected with respect to service rates of hosting VNFs. The migration behavior changes not only the corresponding link latency of path of s_k, but also the response latency of all related VNFs. Fig. 5 shows load balancer (LB) is migrated

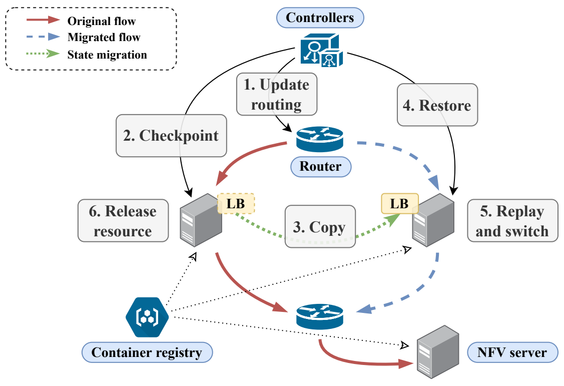

Figure 6: An overview of live migration of a stateful VNF: network traffic re-routing and transfer of VNF states. Container registry distributes images across NFV servers and controllers manage flow control.

and resources in source and target servers are reallocated accordingly.

Let Y_s and Y_t be two lists of VNFs residing on both server v_s and server v_t, respectively. Due to resources released for v_s, service rate of the rest VNFs in v_s may increase. In the following equation, Q_s is the total reduced response latency of all VNFs in source server v_s.

$$Q_s = \sum_{f \in Y_s} \frac{1}{\mu_f - \lambda_f} - \sum_{f \in Y_s \setminus \{f_m\}} \frac{1}{\mu'_f - \lambda_f} \tag{14}$$

Similarly, on the target server v_t, response latency of VNFs may increase. Q_t denotes variance of latency for all VNFs on v_t.

$$Q_t = \sum_{f \in Y_t} \frac{1}{\mu_f - \lambda_f} - \sum_{f \in Y_t \cup \{f_m\}} \frac{1}{\mu'_f - \lambda_f} \tag{15}$$

Hence, the total profit of migrating the VNF f_m, denoted as Q, will be the sum of reduced response latency Q_s, increased response latency Q_t and variance of link latency Q_l after migration. In order to maximize the profit Q, we expect to reduce link latency and response latency by collaboratively finding appropriate VNFs to be migrated and the target server v_t.

$$Q = Q_s + Q_t + Q_l \tag{16}$$

3.4.2. *Cost of migrating a VNF*

VNF migrations will incur extra costs like resource usages and temporary service interruptions. As the number of flows the SFC processes increases, the number of states increases correspondingly, which in turn increases migration time. Besides, service downtime is one of the factors that impair SFC performance. To mitigate such impacts, VNFs can be migrated via live migration, which allows transferring only in-memory states of VNFs (including copying, handling dirty memory pages, resuming, etc., to ensure consistency) rather than the whole VMs to greatly reduce migration time. Fig. 6 shows a basic procedure of migration of a stateful VNF through container live migration [23], including modification of routing for traffic flows and migration of VNF states. Live migration of VMs or containers can constrain the duration of service interruptions to a tolerable level, e.g., in the level of sub-seconds, such that end users are unaware of these changes. Since the focus of this paper is SFC optimization in the edge-core networks, detailed implementations of live migration can be found in [13, 23, 39].

3.4.3. *Impact of SFC positions on VNF migrations*

Jointly combining edge with core network, there are three possible categories of SFC positions: (i) SFC in edge network, an example of which could be ultra-low latency video streaming; (ii) SFC in core network, for instance, online storage service where latency of accessing to the storage can be tolerant; and (iii) hybrid, containing VNFs in both edge and core networks (an example could be a smart surveillance system [4]). The ratio among the three categories of SFC positions is mostly decided by what kinds of and how many services are provided to service customers. Different SFC positions complicate the traffic patterns in edge-core networks. Regarding SFC positions, VNF migration may happen within edge or core network, or between them. In case of VNF migrations between edge and core network, this causes a problem that traffic flows may travel back and forth multiple times if VNF migrations are not properly scheduled, which leads to higher latency.

4. Algorithm Design

In this section, we design two algorithms to solve the SFC migration problem: Profit-driven Heuristic Search (PHS) and Average Utilization Based (AUB) algorithms.

4.1. *Profit-driven Heuristic Search Migration*

A heuristic approach to reduce the average latency is proposed in Algorithm 1. The main idea focuses on selection of appropriate SFCs, VNFs and target servers for migration. First, SFCs that contribute high latency and high resource utilization are identified. Then, for each chosen SFC, victim VNFs and target servers are determined to maximize the profit of migration.

Given a group of deployed SFCs in edge-core networks, these SFCs are evaluated one by one using two performance indicators: total link latency of the SFC path P_k and ratio of amount of resources allocated to the SFC to total capacity of servers where the SFC traverses. As shown in Equation (18), σ_k is defined as an association of both indicators for an SFC, which is used to estimate whether s_k should be considered for further VNF migration. Latency of all SFC paths are first calculated in Equation (17). Then each value is rescaled in the range from 0 to 1 using *min-max* normalization, i.e., the first part of Equation (18). The second part of Equation (18) shows to what extent does the SFC consume available resources along the path: the value 1 means the SFC monopolizes resources of each server. Z_k is a list of NFV servers where VNFs of the SFC s_k are instantiated. The set of SFCs with above-average σ_k among all SFCs will be further analyzed for VNFs migration.

$$\mathbb{W} = \{x \mid x = \sum\nolimits_{e_j \in P_k} W_j, s_k \in \mathbb{S}\} \tag{17}$$

$$\sigma_k = \frac{\mathbb{W}_k - \mathbb{W}_{min}}{\mathbb{W}_{max} - \mathbb{W}_{min}} + \frac{\sum_{f_m \in s_k} A_m}{\sum_{v_i \in Z_k} C_i}, \forall s_k \in \mathbb{S} \tag{18}$$

To choose victim VNFs, it is necessary to analyze whether impacts of VNFs on their original servers are major or not. Like the selection of SFCs, the impact of a VNF is measured by latency of paths connected by the VNF and ratio of required resources R_m to allocated resources A_m. In addition, the more the number of VNFs in a server v_i (i.e., $|Y_i|$) is, the higher probabilities more profit is gained when migrating a VNF out from that server. This factor also contributes to the impacts of the VNF. Therefore, a victim VNF f_m in server v_i is determined according to τ_m in Equation (20), where M_i represents SFC paths the server v_i consecutively connects to. τ_m is a product of the number of VNFs in Y_i and sum of normalized latency and the ratio of R_m to A_m, which evaluates the impacts of VNF f_m on server v_i. VNFs with above-average τ_m are candidates for migration.

$$\mathbb{W}' = \{x \mid x = \sum\nolimits_{e_j \in M_i} W_j, f_m \in s_k\} \tag{19}$$

$$\tau_m = |Y_i| \cdot (\frac{\mathbb{W}'_k - \mathbb{W}'_{min}}{\mathbb{W}'_{max} - \mathbb{W}'_{min}} + \frac{R_m}{A_m}), \forall f_m \in s_k \tag{20}$$

Once a VNF of an SFC is marked as victim, it will be migrated to another server (i.e., target server), either in edge or core network. The target server can be determined based on the following ideas:

- The target server must have enough resources for initializing a new instance of the migrated VNF. The newly allocated resources should satisfy requirements of both incoming VNF and existing VNFs on the target server.
- To reduce link latency as much as possible, the target server should reside on the shortest path between predecessor and successor of the victim VNF with best efforts. Besides, the order of VNFs in an SFC needs to be considered to prevent traffic flows from traveling between edge and core network back and forth multiple times.
- The target server should be low-utilized, i.e., the capacity of the server is much greater than the sum of VNFs' required resources. Low-utilized servers are likely to instantiate more VNFs of different sizes.

The working illustration of Algorithm 1 is as follows. At the beginning of SFC migration, a small set of SFCs is chosen (Line 1) based on the results of Equation (18). For each chosen SFC, victim VNFs are then evaluated by Equation (20) (Line 3). Algorithm 1 picks the best immediate choice at each stage of VNF migration: selection of target servers follows the standard operating procedures mentioned above (Line 6). Each iteration of VNF migration yields a positive profit that is no less than the predefined threshold $Q_{threshold}$ (Line 7~10), which ensures that each migration of VNF is beneficial. The result will finally converge as the migration process continues. Afterwards, new paths for migrated SFCs will be constructed (Line 13) and used to compute objective value (Line 14) as defined in Equation (11).

Algorithm 1 Profit-driven Heuristic Search Migration

```
Input: $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, network; $\mathbb{S}$, a set of SFCs; $P$, a set of paths
    of SFCs; $Q_{threshold}$; $\delta_{threshold}$.
Output: Positions of migrated VNFs and a set of SFC paths.
 1: $\mathbb{S}' \leftarrow$ Choose SFCs from $\mathbb{S}$ by Equation (18)
 2: for $s_k \in \mathbb{S}'$ do
 3:     $\mathbb{F}' \leftarrow$ Mark victim VNFs of SFC $s_k$ by Equation (20)
 4:     for $f_m \in \mathbb{F}'$ do
 5:         $v_s \leftarrow$ Source server of VNF $f_m$
 6:         $v_t \leftarrow \arg\max_{v_i \in \mathbb{H}_m} Q_t$
 7:         $Q \leftarrow$ Compute profit according to Equation (16)
 8:         if $Q \geq Q_{threshold}$ then
 9:             Migrate $f_m$ from $v_s$ to $v_t$            ▹ VNF migration
10:         end if
11:     end for
12: end for
13: $P' \leftarrow$ Path construction for $\mathbb{S}$
14: $\delta \leftarrow$ Compute objective value with $(\mathbb{G}, \mathbb{S}, P, P')$ according
    to Equation (11)
15: if $\delta \geq \delta_{threshold}$ then
16:     return Positions of migrated VNFs and a set of SFC
        paths
17: else
18:     Undo all changes                               ▹ Migration failed
19: end if
```

If the value is greater than or equal to $\delta_{threshold}$, all changes are committed and a set of new paths will be returned (Line 15~16). Otherwise, the algorithm does not perform any SFC migration (Line 17~18). $\delta_{threshold}$ is used to avoid too frequent SFC migrations, especially when considering network dynamics.

To evaluate time complexity for Algorithm 1, we analyze the worst case of the algorithm: the shortest path needs to be calculated when finding the target server for each victim VNF (Line 6). It is known that time complexity to find the shortest path with Dijkstra algorithm using priority queue is $O((|\mathbb{V}| + |\mathbb{E}|) \cdot \log|\mathbb{V}|)$. Thus, the time complexity for Algorithm 1 is: $O(|\mathbb{S}| \cdot |\mathbb{F}| \cdot (|\mathbb{V}| + |\mathbb{E}|) \cdot \log|\mathbb{V}|)$. Besides, Algorithm 1 uses extra memory space to store intermediate values when determining SFCs and victim VNFs, the space complexity for Algorithm 1 is: $O(|\mathbb{S}| \cdot |\mathbb{F}|)$.

4.2. Average Utilization Based Migration

The main focus of Algorithm 1 is to make VNF migration decisions from the perspective of SFC. In this section, another approach based on average utilization of servers is provided with more flexibility.

There are two main observations when intending to reduce average latency: (i) SFCs should be placed on the paths where link latency is as optimal as possible; (ii) processing capacity of VNFs can be improved to reduce response latency by allocating more resources to those VNFs. Inspired by the second observation, all NFV servers in the whole network are divided into two groups: under-utilized servers and over-utilized servers. Equation (21) evaluates all NFV servers by resource utilization. An under-utilized server is a server where total required resources of

Algorithm 2 Average Utilization Based Migration

```
Input: $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, network; $\mathbb{S}$, a set of SFCs; $P$, a set of paths of SFCs; $\delta_{threshold}$.
Output: Positions of migrated VNFs and a set of SFC paths.
 1: $P' \leftarrow \varnothing$
 2: $\mathbb{V}' \leftarrow$ Choose over-utilized servers from $\mathbb{V}$ by Equation (21)
 3: $\mathbb{V}'' \leftarrow \mathbb{V} \setminus \mathbb{V}'$                ▷ Under-utilized servers
 4: for $v_s \in \mathbb{V}'$ do
 5:     $\mathbb{F}' \leftarrow$ Get victim VNFs from $v_s$ by Equation (20)
 6:     for $f_m \in \mathbb{F}'$ do
 7:         $v_t \leftarrow \arg\max_{v_i \in \mathbb{V}'' \cap \mathbb{H}_m} Q_t$
 8:         Migrate $f_m$ from $v_s$ to $v_t$                ▷ VNF migration
 9:         $P' \leftarrow$ Path construction for $\mathbb{S}$
10:     end for
11: end for
12: $\delta \leftarrow$ Compute objective value with $(\mathbb{G}, \mathbb{S}, P, P')$ according to Equation (11)
13: if $\delta \geq \delta_{threshold}$ then
14:     return Positions of migrated VNFs and a set of SFC paths
15: else
16:     Undo all changes                ▷ Migration failed
17: end if
```

all VNFs occupy less than half of the capacity of the server, i.e., $\phi_i < 1/2$. On the contrary, ϕ_i in over-utilized servers is greater than 1/2. To reduce the average latency, VNFs can be migrated from over-utilized servers to under-utilized ones on paths with lower latency.

$$\phi_i = \frac{\sum_{f_m \in Y_i} R_m}{C_i}, \forall v_i \in \mathbb{V} \tag{21}$$

The working illustration of Algorithm 2 is as follows. Both under-utilized and over-utilized servers are first identified according to Equation (21) (Line 2~3). For each over-utilized server, one or more VNFs are selected as victim VNFs based on Equation (20) (Line 5). Afterwards, optimal servers are chosen from under-utilized servers for these victim VNFs using policy of choosing target servers aforementioned in Algorithm 1 (Line 7). It is remarkable that there is path construction after each VNF migration (Line 8~9). The rest of Algorithm 2 is to decide whether to commit changes made by previous decisions and is the same with that of Algorithm 1 (Line 12~17).

Algorithm 2 is based on average utilization. We assume that servers that host VNFs are over-utilized, then the worst case is all VNFs are relocated (Line 7~8) and a new shortest path are constructed for each VNF (Line 9). Hence the time complexity for this algorithm will be: $O(|\mathbb{V}|\cdot|\mathbb{F}|\cdot(|\mathbb{V}|+|\mathbb{E}|)\cdot\log|\mathbb{V}|)$. Algorithm 2 focuses on finding out under-utilized and over-utilized servers and victim VNFs, and then initializes them as array lists, the space complexity for Algorithm 2 is: $O(|\mathbb{V}| \cdot |\mathbb{F}|)$.

5. Performance Evaluation

In this section, extensive simulations are conducted to evaluate the performance of the proposed algorithms.

5.1. Experiment Setup

All codes of the proposed algorithms are written in Java. And all simulations are conducted on a server with 8-core CPU Apple M1 chip and 8GB unified memory. Each simulation consists of a batch of 10,000 random runs with the same configuration of network topology.

Network topology. Several network topologies are used to construct the edge-core networks. BsonetEurope from the Internet Topology Zoo [40] is used as one network topology. Besides, random network topologies are created to simulate different scales of the network, e.g., small, medium, and large. The random network topology is similar to the graph depicted in the Fig. 1, including one edge network and one core network. Servers in edge and core networks have different amounts of computing resources, where capacity scales from 10 to 30 units. All servers are set up according to configurations defined in [21]: one CPU core equals 150 units of resources and one unit of resource can handle 64 bytes packets at 10 *kpps*. Link latency (i.e., UE to edge network: 1-10 *ms*, UE to core network: 10-15 *ms*) and bandwidth (sufficient) between any two connected servers are set according to the actual situation as well.

VNFs & SFCs. A set of commonly-deployed VNFs (including network address translator, load balancer, firewall, deep packet inspector, etc.) are simulated. Each VNF has different requirements of resources, which is also measured in the way computing resources of servers are defined. There is a constraint that total resources consumed by all deployed SFCs must not exceed network capacity. Each SFC is assigned an ingress node and an egress node in a stochastic manner. Random numbers of VNFs are selected from the set and are chained together in a specific order to form an SFC. SFCs with different numbers of VNFs, i.e., lengths, are evaluated.

Traffic flows. Different numbers of traffic flows are assigned to each SFC to simulate users requesting SFCs. Flow rate varies from 1 *kpps* to 2 *kpps* per flow. Each SFC is capable of handling 5 to 10 traffic flows so as to assure the requirement on utilization of each VNF, as shown in Equation (2).

Baseline algorithm. To evaluate the feasibility and performance of two proposed algorithms, a state-of-the-art SFC migration algorithm Follow-Me Chain (FMC) [16] is implemented from scratch. FMC solves the SFC embedding problem considering user mobility to maximize user satisfaction by utilizing a range-based depth-first search (DFS) to select proper migration paths. The SFC migration algorithm of FMC chooses nearby servers with sufficient residual resources and evaluates paths similarity with a Jaccard Similarity Coefficient. As a result, if we assume the worst case is that all VNFs are migrated among the N_{max} highest ranking migrating candidates paths, then the time complexity for FMC is therefore given by: $O(|\mathbb{V}|\cdot|\mathbb{F}|\cdot N_{max})$, and the space complexity is: $O(|\mathbb{V}| \cdot |\mathbb{F}|)$ in our implementation. Before SFC migration, all required SFCs need to be successfully deployed in the network. A feasible algorithm for SFC placement [41] is adopted to provide a uniform environment for SFC migration. The main idea of the placement algorithm is: (i) two adjacent matrices, i.e., physical network graph and the SFC request, are computed as the similarity matrix; (ii) given the server

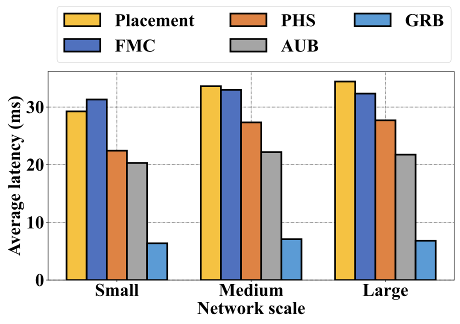

Figure 7: Average latency under different network scales.

resource constraint, SFCs are mapped to the physical network based on the Hungarian method. It can be easily extended and modified to cater to the needs of our experiments and applied to edge-core networks.

Gurobi optimizer. To further analyze the optimal solution to the SFC migration problem, our optimization model is evaluated using a mathematical optimization solver called Gurobi[8] (GRB). Gurobi uses the branch-and-cut algorithm to solve Mixed Integer Program (MIP) models. Since Gurobi takes a very long time to solve the objective function due to the large number of VNFs in large-scale networks, it is necessary to set model attributes and parameters like *MIPGap* and *TimeLimit* for performance improvements. *MIPGap* indicates the relative MIP optimality gap between an upper bound (incumbent) and a lower bound on the optimal objective value. In our average latency minimization model, the upper bound gives the objective of the best-known feasible solution, while the lower bound is the best possible objective. Gurobi will terminate the algorithm and may yield a non-optimal solution if it exceeds *TimeLimit*.

5.2. Results and Analysis

Several metrics are used to evaluate the performance of these algorithms, including migration rate, reduced average latency, execution time and so forth. In all simulations, the maximum resources consumed by every set of SFCs will not exceed the capacity of chosen networks.

Average latency of SFCs. To compare performance of three SFC migration algorithms under different network scales, Fig. 7 shows average latency of one placement algorithm, three migration algorithms and the optimization model solved by the Gurobi solver. It is shown that the proposed algorithms effectively optimize average latency of SFCs and AUB has the best performance compared to the other two algorithms, reducing average latency by up to 12.8 *ms*. Compared to FMC, AUB can achieve up to 36.9% in average latency reduction. Interestingly, FMC incurs extra average latency after migration in the small-sized network. That is because the smaller-sized network offers fewer appropriate paths but FMC must sacrifice existing optimal paths to adapt to user mobility. GRB gives the optimal objective value under all network scales after SFC migration, up to 80.2% average latency reduction.

Fig. 8 gives a comparison about reduced average latency among algorithms in the same network, where the length of SFCs varies from 4 to 8. When the size of SFCs is relatively small, network resources are abundant for these SFCs. Under these circumstances, PHS and AUB behave little differently in minimizing average latency (length of SFCs is 4). When the length of SFCs increases, placement of SFCs results in imbalance of server utilization, of which AUB tries to optimize. Therefore, reduced average latency for AUB increases sharply compared to the other two algorithms. In all cases, FMC can only reduce little average latency, while GRB always has the best performance in average latency reduction, even if either *MIPGap* is set or GRB triggers early termination because *TimeLimit* is reached. Fig. 8 shows for each bar of GRB, there is a gap (i.e., the yellow area on top) between the objective value of the current incumbent and the current objective bound. The optimal objective value always falls into this range.

To find out what contributes to reduction of average latency of SFCs, reduced average latency is further decomposed into link latency and VNF processing latency, as shown in Fig. 9. For the first three algorithms, link latency is the most significant factor that can be optimized, which is achieved by finding the shortest paths for migrated SFCs. Due to SFC latency constraints defined in SLAs, every VNF is allocated enough resources to function normally even though computing resources of a server are evenly shared. Therefore, packet processing can be very fast and VNF processing latency can be low compared to link latency. However, AUB optimizes processing latency by nearly 1/3 of reduced average latency. AUB aims to migrate VNFs from over-utilized servers to under-utilized servers, which increases overall VNF service rates. As a result, VNF processing latency is reduced much more in contrast to the other two algorithms. GRB optimizes more processing latency than the others. Nevertheless, link latency reduction still contributes most to the overall end-to-end latency optimization of the SFC.

Both PHS and AUB can optimize average latency of SFCs and outperforms the comparative algorithm FMC. Fig. 10 shows the normal cumulative distribution function (CDF) for FMC, PHS, AUB and GRB. FMC has nearly 45.6% of migration that cannot be optimized. AUB has higher probabilities to reduce average latency by approximately 2 times than PHS does in peak performance. AUB can achieve reduced average latency by about 26.9 *ms* while PHS stops at most 22.0 *ms*. Overall, GRB still has the best performance compared to the other three algorithms.

Acceptance ratio & migration rate. Migration failure refers to a state that no SFC is migrated after execution of the algorithm, which is potentially caused by algorithm selection. This can be reflected in the acceptance ratio. As Fig. 11 shows, FMC always performs SFC migration blindly no matter what type of networks, hence, 100% success rate of SFC migration. However, the proposed algorithms make smarter choices about whether to migrate SFCs based on calculated profits and objective values. Compared to PHS, AUB tends to have a higher success rate because the strict profit-based migration condition is relaxed. As the network scales up, PHS and AUB have more opportunities to optimize paths and choose target servers, therefore both have

[8]Gurobi 9.5 release, https://www.gurobi.com

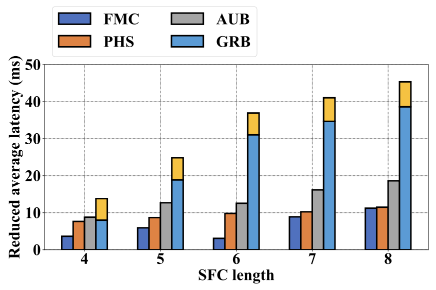

Figure 8: Reduced average latency varying along with SFC length. (The yellow area on top of GRB bar is the relative MIP optimality gap.)

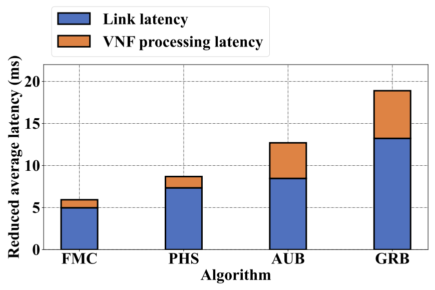

Figure 9: Reduced average latency: link latency vs. VNF processing latency (SFC length = 5).

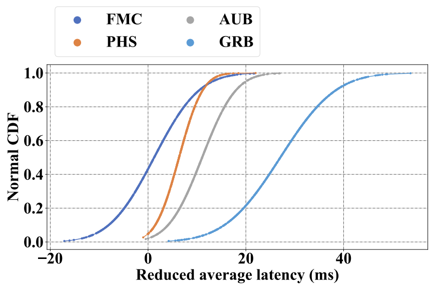

Figure 10: Normal CDF for reduced averaged latency.

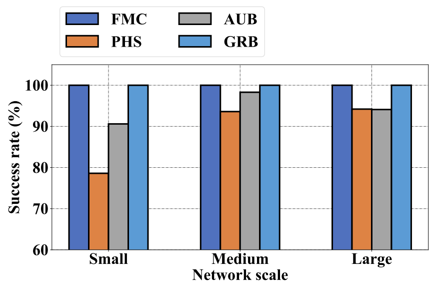

Figure 11: Acceptance ratio under different network scales.

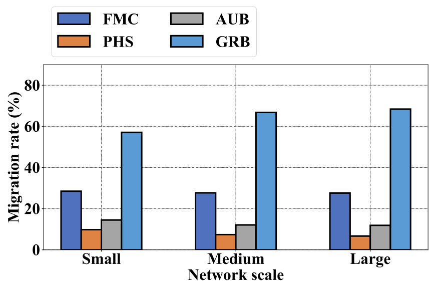

Figure 12: Migration rate upon successful SFC migrations in different networks.

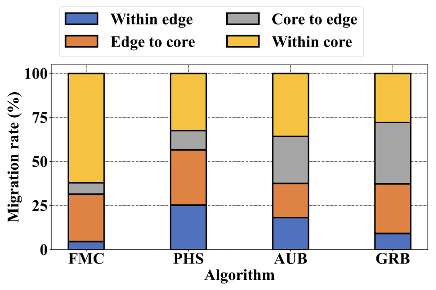

Figure 13: VNF migration within or between edge and core networks.

higher acceptance ratio. GRB also has a 100% success rate of SFC migration. This is because the initial SFC placement algorithm does not achieve the optimal goal of minimizing average latency, and GRB triggers VNF migration for each scenario.

Upon successful SFC migrations, migration rate is used to evaluate how many VNFs are intentionally migrated per run per algorithm. This metric is important because frequent VNF migrations introduce service interruption and overhead, leading to network resources wastage. Thus, fewer VNF migrations are expected to achieve better performance, i.e., lower average latency after migration. Fig. 12 shows the corresponding migration rate for each algorithm. Obviously, FMC migrates over 20% VNFs, about 2.2 times more on average than that of AUB. PHS chooses victim VNFs aggressively based on Equation (18) and (20) and decides VNF migration following rigid rules: profit of migrating a VNF is greater than the predefined threshold. Hence, PHS performs less VNF migration by 6.7% than AUB does. At this point, PHS optimizes latency by nearly 19.5%. Nevertheless, both PHS and AUB migrate VNFs at an acceptable rate. Fig. 12 also demonstrates that the overall migration rate (up to 68.4%) for GRB is the highest among other algorithms, which implies most VNFs are relocated for better performance. However, this solution is not feasible in actual SFC migration, but it helps make better decisions for planning SFC placement through prescriptive analytics.

Furthermore, migration of VNFs among edge and core networks are also investigated. Fig. 13 provides the distribution of migration in different locations. FMC performs most migrations within core network and inclines to migrate VNFs from edge to core network. This reflects that FMC has poor performance in reducing average latency as end users must experience higher service delay due to longer distance. PHS focuses on migration within edge network but there are still VNFs being migrated to core network. The reason is PHS tries to select resource-abundant servers as target servers when making profit-based choices. The distribution in the case of AUB is relatively even because AUB treats VNFs equally and migrates them to nearby suitable servers. Besides, about one quarter of migrated VNFs take place in offloading from core to edge network, which boosts the reduction of average latency of SFCs. GRB mainly migrates VNFs between edge network and core network. This makes sense as the optimal solution may be in this situation: GRB not only optimizes link latency by offloading most VNFs from core to edge network, but also optimizes VNF processing latency by reasonably allocating resources, i.e., migrating VNFs from edge to core network.

Algorithm efficiency. Table 2 shows execution time of all

Table 2: Average execution time (*seconds*)

Network scale	FMC	PHS	AUB	GRB
Small	0.213	0.073	0.420	5.373
Medium	0.592	0.221	0.809	127.235
Large	1.362	0.896	2.149	1.162e+04*

* There exist situations where the result cannot be obtained within the predefined *TimeLimit* (see Table 3).

Table 3: GRB optimization runtime (*seconds*)

SFC length	Small	Medium	Large
4	0.092	8.282	177.735
5	0.435	35.846	377.126
6	4.006	40.873	1.246e+03
7	9.605	127.501	4.466e+04
8	12.728	423.674	-

algorithms upon successful migrations. In all simulations with different network scales, the average execution time of PHS is always the fastest (i.e., up to 2.9 times faster than FMC). The execution of AUB is slightly slower since there are complex operations such as full path reconstruction for every migrated VNFs. Even though in the large-sized network, PHS and AUB solves the SFC migration problem within seconds, proving high efficiency of the algorithms. GRB is multiple orders of magnitude slower than the others. To investigate deeper for GRB, Table 3 shows GRB optimization runtime for SFC migration with different network scales and SFC lengths. The attribute *MIPGap* is set to 0.7 for the medium and large-sized networks. When either the network scales up or the number of VNFs increases, GRB takes longer to solve the optimization model. In the case where SFCs of length 8 are deployed in the large-sized network, GRB fails to achieve an effective result as it exceeds the predefined *TimeLimit*. Apparently, GRB offers optimal solutions to reduce average latency of SFCs, but it must trade time for optimality.

6. Future Works

This paper mainly focuses on SFC migration in the context of edge-core networks such that average latency of all SFCs is optimized. To adapt to network dynamics, both PHS and AUB algorithms can perform periodic checks of the possibility of SFC migration. However, there are several issues that need to be considered for future works:

Shareable VNFs. In practice, a single VNF instance can be allowed to be shared among multiple SFCs. Reusing VNFs can not only maximize resource utilization but also make management of global VNF states easier. Nevertheless, migrating a shareable VNF for one SFC will be a hazard as it potentially violates consistency of other SFCs.

Multipath for SFCs. Under the circumstances that service requests spike during a particular period, network service providers may schedule new SFCs to satisfy incoming requests by scaling up or scaling out VNF instances. Additionally, traffic flows may be re-routed for the purposes of load balance and increasing availability. Therefore, an SFC can have multiple paths regardless of ingress and egress.

Inconsistency with flow changes. Some types of VNFs may involve activities affecting rate of flows such as rate control, traffic modification, etc. This would introduce great challenges when applying the open Jackson network model and more issues remain to be explored further.

7. Conclusions

In this paper, we, for the first time, applied open Jackson networks to model SFCs and schedule migration of SFCs in edge-core networks. The schedule of SFCs aims to optimize average latency of all deployed SFCs as well as reasonably fulfill all requirements that are predefined in policies according to the SLAs. The proposed algorithms Profit-driven Heuristic Search and Average Utilization Based migration efficiently solved the *Latency-aware Edge-Core SFC Migration* problem. The simulations mainly focused on evaluating reduced average latency, migration rate and algorithm efficiency among all algorithms. The results showed that both schemes are feasible: PHS optimizes average latency of all deployed SFCs by nearly 19.5% with 6.7% migration rate, while AUB can reduce average latency by up to 36.9%, which outperforms the state-of-the-art algorithm.

Acknowledgments

This work has been partially supported by National Natural Science Foundation of China (NSFC) No. 62172189 and 61772235; Natural Science Foundation of Guangdong Province No. 2020A1515010771; Science and Technology Program of Guangzhou No. 202002030372; The UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/2 and EP/P004024/1, and InnovateUK grant 106199-47198.

References

[1] R. Gupta, D. Reebadiya, S. Tanwar, 6G-enabled edge intelligence for ultra-reliable low latency applications: Vision and mission, Computer Standards & Interfaces 77 (2021) 103521.

[2] I. Alam, K. Sharif, F. Li, Z. Latif, M. Karim, S. Biswas, B. Nour, Y. Wang, A survey of network virtualization techniques for Internet of Things using SDN and NFV, ACM Computing Surveys (CSUR) 53 (2020) 1–40.

[3] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, C. Verikoukis, Online VNF lifecycle management in an MEC-enabled 5G IoT architecture, IEEE Internet of Things Journal 7 (2019) 4183–4194.

[4] I. Sarrigiannis, E. Kartsakli, K. Ramantas, A. Antonopoulos, C. Verikoukis, Application and network VNF migration in a MEC-enabled 5G architecture, in: IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), IEEE, 2018, pp. 1–6.

[5] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, J. Uttaro, Service Function Chaining Use Cases in Mobile Networks, Internet-Draft draft-ietf-sfc-use-case-mobility-09, Internet Engineering Task Force, 2019. URL: `https://datatracker.ietf.org/doc/html/draft-ietf-sfc-use-case-mobility-09`, work in Progress.

[6] P. K. Thiruvasagam, A. Chakraborty, A. Mathew, C. S. R. Murthy, Reliable placement of service function chains and virtual monitoring functions with minimal cost in softwarized 5G networks, IEEE Transactions on Network and Service Management (2021).

[7] D. Zhao, J. Ren, R. Lin, S. Xu, V. Chang, On orchestrating service function chains in 5G mobile network, IEEE Access 7 (2019) 39402–39416.

[8] C. Zhang, X. Wang, A. Dong, Y. Zhao, F. Li, M. Huang, The intelligent multi-domain service function chain deployment: Architecture, challenges and solutions, International Journal of Communication Systems 34 (2021) e4665.

[9] Y. Zeng, Z. Shi, Z. Wu, VNF placement and routing algorithm for energy saving and QoS guarantee, in: Proceedings of the 9th International Conference on Computer Engineering and Networks (CENet), Springer, 2021, pp. 911–919.

[10] N. Kiran, X. Liu, S. Wang, C. Yin, VNF placement and resource allocation in SDN/NFV-enabled MEC networks, in: IEEE Wireless Communications and Networking Conference Workshops (WCNCW), IEEE, 2020, pp. 1–6.

[11] Y. Alahmad, A. Agarwal, VNF placement strategy for availability and reliability of network services in NFV, in: IEEE 6th International Conference on Software Defined Systems (SDS), IEEE, 2019, pp. 284–289.

[12] B. K. Umrao, D. K. Yadav, Algorithms for functionalities of virtual network: A survey, The Journal of Supercomputing (2021) 1–72.

[13] J. Zhang, L. Li, D. Wang, Optimizing VNF live migration via para-virtualization driver and QuickAssist technology, in: IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.

[14] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, Towards studying service function chain migration patterns in 5G networks and beyond, in: IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[15] D. Zhao, G. Sun, D. Liao, S. Xu, V. Chang, Mobile-aware service function chain migration in cloud–fog computing, Future Generation Computer Systems 96 (2019) 591–604.

[16] Y.-T. Chen, W. Liao, Mobility-aware service function chaining in 5G wireless networks with mobile edge computing, in: IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.

[17] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Communications Surveys & Tutorials 19 (2017) 2322–2358.

[18] J. R. Jackson, Networks of waiting lines, Operations research 5 (1957) 518–521.

[19] D. Li, P. Hong, K. Xue, J. Pei, Virtual network function placement and resource optimization in NFV and edge computing enabled networks, Computer Networks 152 (2019) 12–24.

[20] L. Cui, F. P. Tso, S. Guo, W. Jia, K. Wei, W. Zhao, Enabling heterogeneous network function chaining, IEEE Transactions on Parallel and Distributed Systems 30 (2018) 842–854.

[21] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, T. Wang, Joint optimization of chain placement and request scheduling for network function virtualization, in: IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 731–741.

[22] D. Harutyunyan, R. Behravesh, N. Slamnik-Kriještorac, Cost-efficient placement and scaling of 5G core network and MEC-enabled application VNFs, in: IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2021, pp. 241–249.

[23] K. Govindaraj, A. Artemenko, Container live migration for latency critical industrial applications on edge computing, in: IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, IEEE, 2018, pp. 83–90.

[24] H. Hawilo, M. Jammal, A. Shami, Orchestrating network function virtualization platform: Migration or re-instantiation?, in: IEEE 6th International Conference on Cloud Networking (CloudNet), IEEE, 2017, pp. 1–6.

[25] R. Chen, H. Lu, Y. Lu, J. Liu, MSDF: A deep reinforcement learning framework for service function chain migration, in: IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2020, pp. 1–6.

[26] D. Chemodanov, P. Calyam, F. Esposito, R. McGarvey, K. Palaniappan, A. Pescapé, A near optimal reliable orchestration approach for geo-distributed latency-sensitive SFCs, IEEE Transactions on Network Science and Engineering 7 (2020) 2730–2745.

[27] S. Guo, Y. Dai, S. Xu, X. Qiu, F. Qi, Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT, IEEE Internet of Things Journal 7 (2019) 6010–6022.

[28] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al., MEC in 5G networks, ETSI White Paper 28 (2018) 1–28.

[29] R. Bassoli, F. Granelli, S. T. Arzo, M. Di Renzo, Toward 5G cloud radio access network: An energy and latency perspective, Transactions on Emerging Telecommunications Technologies 32 (2021) e3669.

[30] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, J. P. Jue, Guaranteed-availability network function virtualization with network protection and VNF replication, in: IEEE Global Communications Conference (GLOBECOM), IEEE, 2017, pp. 1–6.

[31] S. Agarwal, F. Malandrino, C.-F. Chiasserini, S. De, Joint VNF placement and CPU allocation in 5G, in: IEEE Conference on Computer Communications (INFOCOM), IEEE, 2018, pp. 1943–1951.

[32] J. Halpern, C. Pignataro, Service function chaining (SFC) architecture, RFC 7665, RFC Editor, 2015.

[33] O. Soualah, M. Mechtri, C. Ghribi, D. Zeghlache, An efficient algorithm for virtual network function placement and chaining, in: IEEE 14th Consumer Communications & Networking Conference (CCNC), IEEE, 2017, pp. 647–652.

[34] P. J. Burke, The output of a queuing system, Operations research 4 (1956) 699–704.

[35] B. Melamed, Characterizations of Poisson traffic streams in Jackson queueing networks, Advances in Applied probability (1979) 422–438.

[36] J. D. Little, S. C. Graves, Little's law, in: Building intuition, Springer, 2008, pp. 81–100.

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

[38] T. He, A. N. Toosi, R. Buyya, SLA-aware multiple migration planning and scheduling in SDN-NFV-enabled clouds, Journal of Systems and Software (2021) 110943.

[39] T. V. Doan, G. T. Nguyen, H. Salah, S. Pandi, M. Jarschel, R. Pries, F. H. Fitzek, Containers vs virtual machines: Choosing the right virtualization technology for mobile edge cloud, in: IEEE 2nd 5G World Forum (5GWF), IEEE, 2019, pp. 46–52.

[40] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The internet topology zoo, IEEE Journal on Selected Areas in Communications 29 (2011) 1765–1775.

[41] M. Wang, B. Cheng, W. Feng, J. Chen, An efficient service function chain placement algorithm in a MEC-NFV environment, in: IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.