Online Adaptive Interference-aware VNF Deployment and Migration for 5G Network Slice

Qixia Zhang, Fangming Liu*, *Senior Member, IEEE,* and Chaobing Zeng

***Abstract*—Based on network function virtualization (NFV) and software defined network (SDN), *network slicing* is proposed as a new paradigm for building service-customized 5G network. In each network slice, service-required virtual network functions (VNFs) can be flexibly deployed in an on-demand manner, which will support a variety of 5G use cases. However, due to the real-time network variations and diverse performance requirements among different 5G scenarios, online adaptive VNF deployment and migration are needed to dynamically accommodate to service-specific requirements. In this paper, we first propose a time-slot based 5G network slice model, which jointly includes both edge cloud servers and core cloud servers. Since VNF consolidation may cause severe performance degradation, we adopt a demand-supply model to quantify the VNF interference. To achieve our objective—maximizing the total reward of accepted requests (i.e., the total throughput minus the weighted total VNF migration cost), we propose an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) for real-time VNF deployment and cost-efficient VNF migration in a 5G network slice, where an Adaptive Interference-aware Algorithm (AIA) is proposed as OLAIA's core function for placing a given set of requests' VNFs with maximized total throughput. Through trace-driven evaluations on two typical 5G network slices, we demonstrate that OLAIA can efficiently handle the real-time network variations and the VNF interference when deploying VNFs for real-time arriving requests. In particular, OLAIA improves the total reward by 22.18% in the autonomous driving scenario and by 51.10% in the 4K/8K HD video scenario, as compared with other state-of-the-art solutions.**

***Index Terms*—Network Function Virtualization, 5G Network Slice, Performance Interference, Online Deployment and Migration.**

I. Introduction

EXPECTED to meet the demands of ultra-high speed, ultra-low latency, and high connection density, 5G network is envisioned to be a multi-service network architecture supporting a wide range of vertical use cases, such as massive Internet of things (IoT), remote machinery, autonomous driving, and virtual reality (VR) [1]. However, the current one-size-fits-all evolved packet core (EPC) network gradually becomes inefficient to handle diverse service requirements and numerous device types [2]. This impels the progress on updating the network architecture with the capabilities of on-demand networking and flexible service deployment for various scenarios.

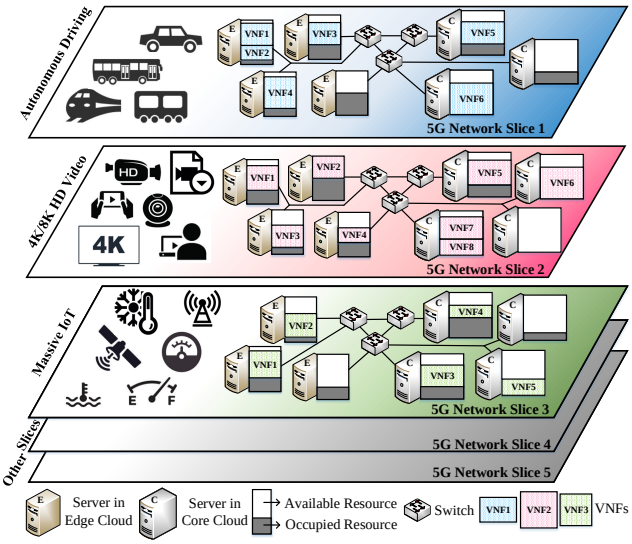

Fig. 1: The 5G network slice infrastructure. In each 5G network slice, it shows a possible VNF placement solution according to the service-specific performance requirements.

Leveraging emerging technologies of network function virtualization (NFV) [3] and software defined network (SDN) [4], *network slicing* is proposed as a new paradigm for building *service-customized* 5G network [1], [5]. Each network slice is composed of virtual resources, independent topology, flow of requests, and service-required network functions (NFs, also known as middleboxes [6], e.g., firewalls, WAN optimizers and load balancers). Thanks to NFV, NFs can be implemented as software-defined virtual network function (VNF) instances running on commercial-off-the-shelf (COTS) servers [7], [8], supporting on-demand service provisioning in each 5G network slice. By transforming the one-size-fits-all service manner to the one-size-per-service manner, each network slice can perform as an end-to-end logical "dedicated network" over the same underlying network, customized for a specific 5G use case. Fig. 1 depicts an example of the 5G network slice infrastructure, including three typical 5G use cases.

Nevertheless, as illustrated in Table I [9], different 5G scenarios have a multiplicity of performance requirements in terms of latency, bandwidth, security, etc. Since NFV decouples the NFs from the dedicated underlying hardware, NFs are no longer restricted with fixed locations [10]. Thus,

This work was supported in part by the NSFC under Grant 61722206 and 61761136014 and 392046569 (NSFC-DFG) and 61520106005, in part by National Key Research & Development (R&D) Plan under grant 2017YFB1001703, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, and in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals. (Corresponding author: Fangming Liu)

Q. Zhang, F. Liu, and C. Zeng are with the National Engineering Research Center for Big Data Technology and System, the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. E-mail: {zhangqixia427, fmliu}@hust.edu.cn, bingzizeng@gmail.com.

TABLE I: 5G Scenarios & Performance Requirements.

Scenario	Latency	Bandwidth	Reliability	Connectivity Density	Security	Mobility
Autonomous Driving	$\leq$ 1 ms	10 Mbps	99.999%	-	High	0 $\sim$ 500 km/h
Industrial Machinery	$\leq$ 1 ms	50 kbps	99.999%	-	High	0 $\sim$ 10 km/h
Remote Surgery	$\leq$ 1 ms	10 Mbps	99.999%	-	High	0 $\sim$ 100 km/h
4K/8K HD Video	$\leq$ 100 ms	$\geq$ 200 Mbps	-	200 $\sim$ 2,500/km^2	-	0 $\sim$ 100 km/h
Mass Gathering	$\leq$ 10 ms	$\geq$ 50 Mbps	-	150,000/km^2	-	0 $\sim$ 10 km/h
Office Automation	$\leq$ 10 ms	$\geq$ 1 Gbps	99.999%	-	High	0 $\sim$ 10 km/h

NFV provides an opportunity to determine how to flexibly deploy the service-required VNFs and how to dynamically migrate some VNFs in each 5G network slice, so that the service-specific requirements can be satisfied and quality of service (QoS) can be further enhanced.

However, deploying VNFs in end-to-end 5G network slice is different from solely deploying VNFs in core network (e.g., core cloud or datacenters) or in edge network (e.g., edge cloud, edge datacenters or cloudlet). Because in an end-to-end 5G network slice, we have to consider placing VNFs in both edge cloud servers (with limited resource capacity and low latency) and core cloud servers (with relatively sufficient resource capacity and high latency) so as to satisfy some strict QoS requirements. For example, as depicted in Fig. 1, autonomous driving network slice needs ultra-low latency and high reliability, which requires to place more VNFs in edge cloud servers. 4K/8K HD video network slice needs high speed and throughput, which requires resource-sufficient core cloud servers and bandwidth-sufficient links to deploy the required VNFs. Even though there has been some progress tackling the VNF placement problem in 5G [11]–[16], their solutions usually focus on placing VNFs in either core cloud servers or edge cloud servers, but not in both types of servers in end-to-end 5G network slices. Besides, network traffic and state typically exhibit great variations due to stochastic arrival of requests [8]; thus, online VNF deployment and migration are both needed to adapt to real-time network variations.

In fact, VNFs are sometimes consolidated on the same server for power conservation, reduction on communication latency, etc., which is called *VNF consolidation* [17]. For example, in Fig. 1, VNF1 and VNF2 are consolidated on the first edge cloud server in the autonomous driving network slice. VNF consolidation can cause severe performance degradation in terms of throughput and latency [18], [19], which is known as *VNF interference*. The throughput even degrades from 12.36% to 50.30% as more VNFs are consolidated on the same server [18]. However, as far as we know, none of the existing works have captured the VNF interference when making VNF deployment and migration decisions, especially for some QoS-strict 5G network slices (e.g., latency-sensitive autonomous driving and throughput-sensitive 4K/8K HD video).

Therefore, differing from previous works, we firstly propose a time-slot based 5G network slice model, which includes both edge cloud servers and core cloud servers. We define the real-time server and link resource utilization (e.g., CPU [20], memory, and bandwidth), real-time network state and real-time arriving requests. Since a service request is typically processed in a service function chain (SFC) [21], [22], which is a set of VNFs that have to be orderly-executed based on a VNF forwarding graph (VNF-FG) [11], we model the VNF-FGs as one-ingress directed graphs to represent the general cases. Next, we conduct a set of empirical evaluations to find out the key factors that reflect the VNF interference, and then we adopt a *demand-supply model* [23] to quantify the VNF interference in terms of degraded throughput.

We formulate the joint VNF deployment and migration problem in 5G network slice as an online optimization problem, which is proved NP-hard. To achieve our objective—maximizing the total reward of accepted requests (i.e., the total throughput minus the weighted total VNF migration cost), we propose an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) for real-time VNF deployment and cost-efficient migration in a 5G network slice. We also design an Adaptive Interference-aware Algorithm (AIA) as OLAIA's core function for placing a given set of requests' VNFs with maximized total throughput. In particular, OLAIA has four major technical insights: (1) adaptive VNF placement in both edge cloud servers and core cloud servers considering each 5G network slice's QoS requirements; (2) quantification of the VNF interference based on the demand-supply model, and relieving the VNF interference by reducing the possibility of VNF consolidation in the same server; (3) real-time VNF deployment and update of network states to handle real-time arriving requests and network variations; and (4) joint cost-efficient VNF migration and redeployment of rejected requests with minimized migration cost and service-customized migration frequency. We give theoretical analyses of both AIA and OLAIA in terms of optimality, competitive ratio, and algorithm complexity. Extensive evaluations on two typical 5G network slices (i.e., autonomous driving and 4K/8K HD video) verify OLAIA's superior performance in maximizing the total reward when deploying and migrating VNFs for real-time arriving requests. The main contributions are as follows:

- We construct a time-slot based 5G network slice model for online VNF deployment and migration, which includes both edge cloud servers and core cloud servers, real-time network state, real-time server and link resource utilization (e.g., CPU, memory, and bandwidth) and real-time arriving requests with their complex VNF-FGs. In particular, we adopt a demand-supply model to quantify the VNF interference in terms of degraded throughput.

- To achieve our objective—maximizing the total reward of accepted requests (i.e., the total throughput minus the weighted total VNF migration cost), we propose an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) for real-time VNF deployment and cost-efficient VNF migration in a 5G network slice,

where the Adaptive Interference-aware Algorithm (AIA) is proposed as OLAIA's core function for placing a given set of requests' VNFs with maximized total throughput. In particular, OLAIA orderly places VNFs in edge cloud servers and core cloud servers with special considerations on the VNF interference, real-time network variations and QoS requirements, and jointly migrates VNFs and redeploys previously rejected requests with minimized migration cost and service-customized migration frequency.

- Through trace-driven evaluations on two typical 5G network slices, the experiment results show that AIA can improve the total throughput of accepted requests by 15.02% in the autonomous driving scenario and by 30.52% in the 4K/8K HD video scenario; and OLAIA can efficiently maximize the long-term total reward with a relatively low migration cost, i.e., improving 22.18% and 51.10% total reward in these two 5G use cases, as compared with other state-of-the-art schemes.

II. Related Work

In this section, we first summarize the state-of-the-art VNF placement solutions in core network and then focus on the current efforts that have been paid on VNF placement in edge network and 5G network.

A. VNF Placement in Core Network

Most existing works formulate the VNF placement as an optimization problem with different objective(s), and thus a variety of solutions are proposed, which are generally classified as either exact ones or heuristic ones [7]. It is known that the exact algorithms offer optimal solutions which are largely limited by the network scale, because their execution time grows exponentially with the network size (e.g., [24], [25]); on the contrary, the heuristic algorithms offer near-optimal solutions with short execution time, which are typically not limited by the network scale (e.g., [21], [26]–[28]).

In order to achieve some specific optimization objective(s) (e.g., minimizing the number of used servers or end-to-end latency, or maximizing the total throughput), mathematical programming methods such as integer linear programming (ILP) [29] and mixed ILP (MILP) [30] are generally used. For instance, Moens *et al.* [25] formulate the VNF placement problem as an ILP, which aims to minimize the number of used servers. Li *et al.* [31] also use an ILP model to provision resources for SFCs. They aim at maximizing the total number of accepted requests, meanwhile providing real-time guarantees. Addis *et al.* [27] devise an MILP formulation for the SFC routing optimization problem, which aims at jointly minimizing the maximum network link utilization and the number of CPU cores used by the instantiated VNFs. Lin *et al.* [30] present an MILP model and propose a game theory-based VNF placement approach to reduce the NF deployment cost. Zhang *et al.* [21] jointly optimize the NF chain placement and request scheduling by modeling the NFV network as an open Jackson network and proposing a two-stage heuristic algorithm, which can simultaneously improve the resource utilization efficiency and reduce the average response latency.

Some current works (e.g., [24], [32]–[34]) also utilize the Markov decision process (MDP) model to capture the characteristics of real-time arriving requests. For example, NFVdeep [34] introduces an MDP model to capture the dynamic network state transitions; then it proposes an online policy gradient-based deep reinforcement learning approach to automatically deploy SFCs, which can jointly minimize the operation cost of NFV providers and maximize the total throughput of accepted requests. However, these solutions mainly consider placing VNFs in core network (e.g., centralized datacenters or core cloud), which can not capture some key features of 5G network slice (e.g., end-to-end real-time service provisioning, 5G-specific QoS requirements, and the performance-degraded VNF interference).

B. VNF Placement in Edge Network and 5G Network

Along with the development of *edge computing*, *fog computing*, and *mobile edge computing* (MEC), it is convenient to use the edge network (e.g., edge cloud, edge datacenters and/or cloudlet) to provide a wide range of services in a low-latency and energy-efficient manner [35], [36]. For example, Jin *et al.* [37] formulate the SFC deployment problem as an MILP and propose a novel two-stage latency-aware VNF deployment scheme to jointly optimize the resource utilization of both edge servers and physical links. Finedge [36] is proposed as a dynamic, fine-grained and cost-efficient edge resource management framework for NFV network, which can efficiently handle heterogeneous flows and improve the resource utilization efficiency. Laghrissi *et al.* [38] tackle the VNF placement in a dynamic edge cloud environment (i.e., edge slicing). Cziva *et al.* [39] also solve the dynamic, latency-optimal VNF placement problem at the edge network.

In 5G network, Cao *et al.* [11] propose a two-step method for solving the VNF-FG design and VNF placement for 5G mobile networks, aiming at minimizing the bandwidth consumption. In order to minimize the resource consumption when making delay-aware VNF placement and chaining decisions in 5G, Alleg *et al.* [14] use a mixed integer quadratically constrained programming (MIQCP) formulation and propose a flexible resource allocation model. Alhussein *et al.* [16] also study the joint traffic routing and VNF placement problem in 5G core network by formulating it as an MILP, which considers one-to-many and many-to-one VNF mapping.

In addition, Agarwal *et al.* [13] present a methodology to make joint VNF placement and CPU allocation decisions in 5G and extend their work in [15] to support more vertical 5G services. Jemaa *et al.* [40] jointly optimize the VNF placement and resource provisioning in edge-central carrier cloud infrastructure with queuing and QoS models. However, they have not integrally captured the complex VNF-FGs, 5G-specific QoS requirements, real-time network variations, and the VNF interference in solving the online VNF deployment and migration problem for end-to-end 5G network slices.

In addition, Zhang *et al.* [41] investigate the offline VNF placement problem in 5G network slices and propose a throughput-maximizing heuristic algorithm. Differing from it and other previous works, we deal with the joint optimization

problem of online VNF deployment and cost-efficient migration for 5G network slice. We propose a time-slot model that captures all important features in 5G network slice, including real-time network state and resource utilization (e.g., CPU, memory, and bandwidth), and real-time arriving requests with their complex VNF-FGs. Our proposed online lazy-migration adaptive interference-aware approach OLAIA can improve the average total reward by at least 68% as compared with the offline non-migration algorithm AIA in [41].

III. 5G Network Slice Model and Problem Formulation

In this section, we first introduce the time-slot based 5G network slice model. Next, we conduct a set of empirical evaluations on consolidated VNFs, and then we adopt a *demand-supply model* to capture the VNF interference in terms of degraded VNF throughput. Finally, we propose the mathematical formulation of the online VNF deployment and migration problem in 5G network slice and prove its NP-hardness. Key notations are listed in Table. II.

A. 5G Network Slice Model

Since a 5G network slice is an end-to-end, logically-isolated network, customized for a specific 5G use case, we represent each 5G network slice as an undirected graph $G = (N, L)$, where N is the set of servers (nodes) and L is the set of links. Notably, we use N_c to represent the set of core cloud servers, while N_e represents the set of edge cloud servers. We assume that there are several levels of switches for ensuring the connectivity of the servers. Each link $l \in L$ corresponds to a pair of nodes (n_1, n_2) where $n_1, n_2 \in N, n_1 \neq n_2$.

To deal with the real-time network variations, we assume that the NFV system executes in a time-slot manner [34], where the set of time slots is $T = \{t_1, t_2, ..., t_{|T|}\}$. At each time slot $t \in T$, each server $n \in N$ has an available resource capacity $c_{n,t} = (c^{cpu}_{n,t}, c^{mem}_{n,t})$, representing its quantity of available resources in terms of CPU and memory. Note that other types of resource, such as storage, can also be added in $c_{n,t}$ if necessary. In a typical 5G network slice, core cloud servers $n \in N_c$ have relatively sufficient resource capability but higher response latency than edge cloud servers $n \in N_e$, which are closer to the end-users with relatively limited resource capability but lower response latency. We denote the link latency of $l = (n_1, n_2) \in L$ by t^{lnk}_l, which includes both the propagation delay and transmission delay; while $b_{l,t}$ represents the available bandwidth resource on link $l \in L$ at time slot $t \in T$.

We use $F = \{f_1, f_2, ..., f_{|F|}\}$ to represent the set of service-required VNFs in each 5G network slice. Each service instance of VNF $f \in F$ has a resource demand in terms of CPU and memory, denoted by $d_f = (d^{cpu}_f, d^{mem}_f)$. If a service instance of VNF $f \in F$ is deployed, it can serve requests with a positive service rate. We define the node latency for VNF $f \in F$ by t^n_f, representing the queuing delay and processing delay on node $n \in N$.

In each service-customized 5G network slice, a request $r \in R$ needs to traverse a set of VNFs $F_r = \{f_1, f_2, ..., f_{|r|}\} \subseteq F$, where $r(i) = f_i \in F_r$. The traffic flow of a request is transferred according to its VNF-FG with a non-negative ingress throughput λ_r. The general VNF-FGs are one-ingress directed graphs without loops, including linear chains, split-and-merged graphs, and other complex one-ingress multi-egress directed graphs. Fig. 2 shows an example of a complex VNF-FG of request $r \in R$, $|r| = 7$, and $F_r = \{f_1, f_2, ..., f_7\}$. Note that not only splitting and merging (e.g., VNF4 and VNF5) will change the throughput, some particular VNFs (e.g., VNF2 and VNF3 for decompression and compression) can also change the throughput. Besides, sometimes several replicas of the same VNF are deployed for load balancing

TABLE II: Key Notations.

Symbol	Description
$G = (N, L)$	The underlying network of a 5G network slice
$N = N_c \cup N_e$	The set of servers (nodes), where N_c is the set of core cloud servers and N_e is the set of edge cloud servers
L	The set of links between the servers, $\forall l = (n_1, n_2) \in L, n_1, n_2 \in N$
T	The set of time slots
F	The set of required VNFs in a 5G network slice
R	The set of requests in a 5G network slice
P_r	The set of VNF-FPs of request $r \in R$, where $P_r = \{P^1_r, P^2_r, ...\}$ and each $P^i_r = (p^i_1, p^i_2, ...)$
$A_r = (a_{r(i,j)})_{\|r\|\times\|r\|}$	The adjacent matrix that represents the VNF-FG of request $r \in R$, where $a_{r(i,j)} \in A_r$ refers to the throughput transferring from VNF $r(i) \in F_r$ to VNF $r(j) \in F_r$
$c_{n,t} = (c^{cpu}_{n,t}, c^{mem}_{n,t})$	The available resource capacity of server $n \in N$ at time slot $t \in T$ in terms of CPU and memory
$d_f = (d^{cpu}_f, d^{mem}_f)$	The resource demand of a VNF instance $f \in F$ in terms of CPU and memory
t^n_f	The node latency for handing VNF $f \in F$ placed on node $n \in N$
t^{lnk}_l	The link latency on link $l \in L$
$q^i_{j,r}$	The equivalent throughput between VNF $r(p^i_j) \in F_r$ and $r(p^i_{j+1}) \in F_r$ in VNF-FP $P^i_r \in P_r$
$\alpha_{n,f,t}$	The demand-supply ratio when VNF $f \in F$ is consolidated with other VNF(s) in server $n \in N$ at time slot $t \in T$
$b_{l,t}$	The available bandwidth resource on link $l \in L$ at time slot $t \in T$
λ_r	The ingress throughput of request $r \in R$
τ^{ari}_r	The arriving time of request $r \in R$
τ^{ser}_r	The service time of request $r \in R$
τ^{cst}_r	The time of constructing all VNF-FPs of request $r \in R$
τ^{rsp}_r	The response latency of request $r \in R$
T_r	The response latency limitation for request $r \in R$
$s_{r,t}$	The service state of request $r \in R$, 1 if r is in service at time slot $t \in T$, 0 otherwise
$x^{r(i)}_{n,t}$	1 if VNF $r(i) \in F_r$ of request $r \in R$ is placed on server $n \in N$ at time slot $t \in T$, 0 otherwise
y_r	1 if request $r \in R$ is accepted, 0 otherwise

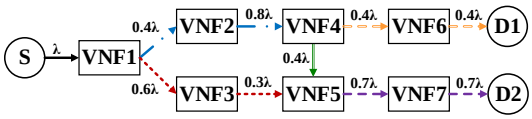

Fig. 2: An example of the VNF-FG of request $r \in R$ with an ingress point S, two egress points D1 and D2, and seven VNFs, where λ is the ingress throughput.

[42], [43] or dynamic scaling [44], we regard each replica as a separate VNF, which can equally divide the throughput.

Based on the VNF-FG of request $r \in R$, we define an $|r| \times |r|$ adjacent matrix A_r to represent the flow direction between every two VNFs, where $a_{r(i,j)} \in A_r$ refers to the throughput of flow transferring from VNF $r(i) \in F_r$ to VNF $r(j) \in F_r$. Each request $r \in R$ has an arriving time τ_r^{ari} and a service time τ_r^{ser}, representing how long request r is expected to be in service if all its VNFs can be placed; otherwise, $\tau_r^{ser} = 0$. Then we define the state of request $r \in R$ by $s_{r,t}$, representing whether request $r \in R$ is in service at time slot $t \in T$:

$$\forall t \in T, r \in R : s_{r,t} = \begin{cases} 1, \tau_r^{ari} \leq t < (\tau_r^{ari} + \tau_r^{ser}), \\ 0, \text{otherwise.} \end{cases} \quad (1)$$

Besides, each request has a response latency, denoted by τ_r^{rsp}, which represents the latency for building a connection between the sending end and receiving end of request $r \in R$. This response latency limitation T_r of request $r \in R$ indicates that if the response latency τ_r^{rsp} exceeds T_r, request r will be rejected temporarily, and r needs to wait for some time slots until the resource and latency constraints are both satisfied.

At the beginning of each time slot $t \in T$, the NFV system will execute the following procedures: scanning all the servers and links, removing timeout requests whose $s_{r,t} = 0$, receiving arriving requests, making VNF deployment and migration decisions, and then updating the network states and accepted requests with $s_{r,t} = 1$.

Finally, we define a binary variable $x_{n,t}^{r(i)}$ to indicate whether VNF $r(i) \in F_r$ of request $r \in R$ is placed on server $n \in N$ at time slot $t \in T$. We also define a binary variable y_r to indicate whether request $r \in R$ is accepted or not, whose mathematical formulation will be introduced later in Sec. III-C.

B. Capturing VNF Interference

Inspired by the work [18], we learn that due to resource contention, performance interference between co-located VNFs is ubiquitous, which can degrade a VNF's throughput by as much as 50.30% as compared with it running in isolation. To measure and quantify the VNF interference, we seek to answer the following two questions: (1) *What is the relationship between the VNF interference and the number of consolidated VNFs?* (2) *What are the key factors that reflect such VNF interference and how to quantify it*?

We address these questions by tentatively evaluating the relationship between the VNF performance and the servers' resource utilization (i.e., CPU and memory) as the number of consolidated VNFs increases on the same server. Each server in this empirical evaluation is configured with an eight-core Intel Xeon E5-2620 v4 2.1 GHz CPU, a 64 GB memory,

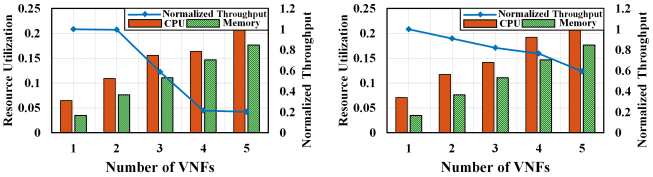

Fig. 3: The resource utilization of servers and the VNF's normalized throughput with different numbers of co-located VNFs.

and two Intel 82599ES 10-Gigabit network interface cards (NICs). In an NFV server, each VNF instance runs on an exclusive virtual machine (VM), and each VM instance is equipped with 1 vCPU core, 2 GB memory and 10 GB disk. We measure two simple VNFs, `Pktstat`[1] and `Snort`[2]. We deploy `Pktstat` and `Snort` on two co-located VMs to perform listening and filtering functions, respectively. We utilize another server to generate and send traffic to each VM. We record the VNF throughput, CPU and memory utilization per second in a minute and repeat each experiment five times. Then we calculate the average values as measurement results.

As illustrated in Fig. 3, we observe that as we scale up the number of consolidated VNFs, the server's resource utilization in terms of CPU and memory grows approximately linearly and the normalized throughput decreases accordingly. In other words, as more VNFs are consolidated on the same server, the resource contention is severer, leading to severer performance degradation in VNF's throughput.

Since VNFs are generally deployed in VMs or containers (e.g., Docker), inspired from the ubiquitous co-located *VM interference* [45], the VNF interference can also be considered as the *mismatch* between the resource "supply" provided by the server and the resource "demand" of co-located VNFs deployed in that server. Thus, we adopt a *demand-supply model* [23] to quantify the VNF interference in terms of degraded throughput. We devise a simple yet effective *demand-supply ratio* $\alpha_{n,f,t}$ to capture the VNF interference when VNF $f \in F$ is consolidated with other VNFs in server $n \in N$ at time slot $t \in T$, which can be represented as follows:

$$\alpha_{n,f,t} = k_0 + k_1 \cdot \frac{D_{n,f,t}^{cpu}}{c_{n,t}^{cpu}} + k_2 \cdot \frac{D_{n,f,t}^{mem}}{c_{n,t}^{mem}}. \quad (2)$$

where k_0, k_1 and k_2 are the coefficients in our statistical linear model, $c_{n,t}^{cpu}$ and $c_{n,t}^{mem}$ are the available CPU and memory resource "supplied" by the server $n \in N$ at time slot $t \in T$; while $D_{n,f,t}^{cpu}$ and $D_{n,f,t}^{mem}$ are the aggregated CPU and memory resource "demand" of all consolidated VNFs (including VNF f) placed in server $n \in N$ at time slot $t \in T$, which can be represented by:

$$\forall n \in N, f \in F, D_{n,f,t}^{cpu} = d_f^{cpu} + \sum_{r \in R} \sum_{i=1}^{|r|} x_{n,t}^{r(i)} \cdot d_{r(i)}^{cpu}. \quad (3)$$

$$\forall n \in N, f \in F, D_{n,f,t}^{mem} = d_f^{mem} + \sum_{r \in R} \sum_{i=1}^{|r|} x_{n,t}^{r(i)} \cdot d_{r(i)}^{mem}. \quad (4)$$

[1] `Pktstat`: a flow monitor to display a real-time summary of packet activity on an interface, https://linux.die.net/man/1/pktstat

[2] `Snort`: a network intrusion prevention system with real-time traffic analysis, https://www.snort.org/

This way, $D^{cpu}_{n,f,t}/c^{cpu}_{n,t}$ and $D^{mem}_{n,f,t}/c^{mem}_{n,t}$ represent the CPU and memory resource utilization, while $\alpha_{n,f,t} \in (0,1]$ indicates the ratio of degraded throughput of VNF f on server n as compared with the throughput when f is exclusively placed on a server. Thus, the egress throughput after executing VNF f should be multiplied by $\alpha_{n,f,t}$. The coefficients k_0, k_1 and k_2 in Eq. (2) are derived from the measurement results in [18], which are different in different NFV platforms (e.g., VM-based or container-based platform). In our model, we firstly use the results in [18] as an initial setup and then construct experiments with different numbers of VNFs running on the same server. We trace the VNF's throughput together with the resource utilization and keep updating and refining the model coefficients for calibration. Eq. (2) can also be extended to other dimensional resources, such as storage and bandwidth.

C. Problem Formulation

Now we formally propose the mathematical formulation of the **online VNF deployment and migration problem in 5G network slice**. We begin with the constraints.

First, at any time slot $t \in T$, any VNF $r(i) \in F$ of a request $r \in R$ can only be either placed on one server or not placed at all, while different service instances of the same VNF will be regarded as different VNFs (VNF replicas) as we explained previously. Thus, we have:

$$\forall t \in T, r(i) \in F : \sum_{n \in N} x^{r(i)}_{n,t} \leq 1. \tag{5}$$

$$\forall t \in T, r(i) \in F, n \in N : x^{r(i)}_{n,t} \in \{0,1\}. \tag{6}$$

Next, at each time slot $t \in T$, the total resource demand of consolidated VNFs on the same server cannot exceed its resource capacity, which can be expressed as:

$$\forall t \in T, n \in N : \sum_{r \in R} \sum_{i=1}^{|r|} x^{r(i)}_{n,t} \cdot d_{r(i)} \leq c_{n,t}. \tag{7}$$

We also want to know whether the throughput of all requests along a link $l \in L$ exceeds its link bandwidth capacity $b_{l,t}$. We first find out all the *VNF Forwarding Paths (VNF-FPs)* of a VNF-FG, which is a set of linear chains from the ingress point to each of the egress points. Since each VNF-FG has only one ingress point and a fixed number of egress points, the set of all VNF-FPs in a VNF-FG is also fixed. Leveraging classic graph searching algorithms such as depth-first search (DFS) [46], we can easily construct the VNF-FP set of a VNF-FG. We define the constructing time of request $r \in R$ as τ^{cst}_r for constructing all of its VNF-FPs by using DFS. Apparently, τ^{cst}_r depends on the scale of a VNF-FG. Based on the survey of VNFs [47] and SFCs [22], typically, there will be no more than 10 VNFs in a VNF-FG, and thus by adopting DFS, τ^{cst}_r can be obtained within a millisecond. We can easily get a set of all VNF-FPs of request $r \in R$ represented by $P_r = \{P^1_r, P^2_r, ...\}$, where each $P^i_r = (p^i_1, p^i_2, ...)$ stores the chain of VNFs $\forall r(p^i_j) \in F_r$ along its VNF-FP, and p^i_j refers to an integer index in request $r \in R$. Assume that $p^i_j = k^i_j \in \{1, 2, ..., |F_r|\}$, then $r(p^i_j) = f_{k^i_j} \in F_r$ and $r(P^i_r) = r(p^i_1, p^i_2, ...) = (f_{k^i_1}, f_{k^i_2}, ...)$. For example, as depicted in Fig. 2, there are three VNF-FPs, i.e., P^1_r, P^2_r, and P^3_r respectively, where $P^1_r = (p^1_1, p^1_2, p^1_3, p^1_4) = (1,2,4,6), P^2_r = (1,2,4,5,7)$, and $P^3_r = (1,3,5,7)$. Thus, for instance, $r(P^1_r) = r(p^1_1, p^1_2, p^1_3, p^1_4) = (f_1, f_2, f_4, f_6)$.

For each $r \in R$, we also define a list of $Q_r = \{Q^1_r, Q^2_r, ...\}$, where each $Q^i_r = (q^i_{1,r}, q^i_{2,r}, ...)$ represents the equivalent split throughput of VNF-FP $P^i_r \in P_r$, i.e., the throughput between each pair of VNFs (VNF $r(p^i_j) \in F_r$ and VNF $r(p^i_{j+1}) \in F_r$) in P^i_r. Based on the assumptions made previously, the throughput can be equally split among multiple VNF-FPs if they share the same pair of VNFs, i.e., VNF $r(p^i_j)$ and VNF $r(p^i_{j+1})$. Thus, the definition of $q^i_{j,r}$ can be represented by:

$$\forall r \in R, i \in [1, |P_r|], j \in [1, |P^i_r| - 1], q^i_{j,r} \in Q^i_r,$$
$$q^i_{j,r} = a_{r(p^i_j, p^i_{j+1})} \cdot \frac{1}{\sum_{v=1}^{|P_r|} \sum_{u=1}^{|P^v_r|} [p^v_u(p^i_j), p^v_{u+1}(p^i_{j+1})]^+}. \tag{8}$$

where $a_{r(p^i_j, p^i_{j+1})}$ is the actual throughput between VNF $r(p^i_j)$ and VNF $r(p^i_{j+1})$, which can be obtained from its adjacent matrix; $[p^v_u(p^i_j), p^v_{u+1}(p^i_{j+1})]^+$ is a function to calculate how many VNF-FPs share the same link between VNF $r(p^i_j)$ and node $r(p^i_{j+1})$, where if $p^v_u = p^i_j$ and $p^v_{u+1} = p^i_{j+1}$, then $[p^v_u(p^i_j), p^v_{u+1}(p^i_{j+1})]^+ = 1$, otherwise it equals to 0. For example, as plotted in Fig. 2, two VNF-FPs P^1_r and P^2_r share the link between VNF1 and VNF2, thus $q^1_{1,r} = a_{r(p^1_1, p^1_2)} \cdot (1/\sum_{v=1}^{|P_r|} \sum_{u=1}^{|P^v_r|} [p^v_u(p^1_1), p^v_{u+1}(p^1_2)]^+) = a_{r(1,2)} \cdot (1/1+1) = 0.4 \cdot (1/2) = 0.2$ and $q^2_{1,r} = 0.2$.

Based on Eq. (8), we can derive the equivalent throughput along each VNF-FP in Fig. 2. For instance, $Q^1_r = (q^1_{1,r}, q^1_{2,r}, q^1_{3,r}) = (0.2, 0.4, 0.4)$, $Q^2_r = (0.2, 0.4, 0.4, 0.35)$ and $Q^3_r = (0.6, 0.3, 0.35)$. As mentioned above, due to the scale of VNF-FGs, typically, $|P_r| \leq 30$ and $|Q_r| \leq 30$, while $|P^i_r| \leq 10$ and $|Q^i_r| \leq 10$.

Note that Eq. (8) does not include the VNF interference, we revise it by $q^{i*}_{j,r}$, where the VNF interference $\alpha^n_{r(p^i_j),t}$ is accumulated along each path. At time slot $t \in T$, we have:

$$q^{i*}_{j,r} = q^i_{j,r} \cdot \frac{\prod_{j=1}^{|P^i_r|} \prod_{n \in N} (\alpha^n_{r(p^i_j),t})^{x^{r(p^i_j)}_{n,t}}}{\prod_{j=1}^{|P^i_r|} \prod_{n \in N} (\alpha^n_{r(p^i_j),t})^{x^{r(p^i_j)}_{n,t} \cdot x^{r(p^i_{j+1})}_{n,t}}}. \tag{9}$$

Since the throughput of all requests along link $l \in L$ should not exceed its bandwidth resource capacity $b_{l,t}$ at time slot $t \in T$, we represent the link resource constraint as below:

$$\forall t \in T, n_1, n_2 \in N, l = (n_1, n_2),$$
$$\sum_{r \in R} \sum_{i=1}^{|r|-1} x^{r(i)}_{n_1,t} \cdot x^{r(i+1)}_{n_2,t} \sum_{v=1}^{|P_r|} \sum_{u=1}^{|P^v_r|} (q^{v*}_{u,r} | p^v_u = i, p^v_{u+1} = i+1) \leq b_{l,t}, \tag{10}$$

where $\sum_{r \in R} \sum_{i=1}^{|r|-1} x^{r(i)}_{n_1,t} \cdot x^{r(i+1)}_{n_2,t}$ calculates all pairs of VNFs placed on $n_1 \in N$ and $n_2 \in N$ for all $r \in R$, and $\sum_{v=1}^{|P_r|} \sum_{u=1}^{|P^v_r|} (q^{v*}_{u,r} | p^v_u = i, p^v_{u+1} = i+1)$ represents the sum of revised throughput from VNF $r(i)$ to $r(i+1)$ in request r considering the VNF interference. In Eq. (10), we assume that two VNFs $r(i)$ and $r(i+1)$ are deployed on nodes n_1 and n_2. If n_1 and n_2 are adjacent nodes, the traffic is directly routed through its physical link l=(n_1,n_2) if it has enough

bandwidth resource $b_{l,t}$; otherwise, if $b_{l,t}$ is not enough, or n_1 and n_2 are nonadjacent nodes, we calculate the aggregated bandwidth of all links from n_1 to n_2 in $b_{l,t}$ (e.g., by using VL2 [48]) and check whether it is larger than the demand. This assumption is also commonly used in some existing works, such as [21], [25], [33], [37]. Besides, we mainly focus on how to deploy and migrate VNFs in this paper, and there is a large number of works focusing on traffic routing between VNFs, such as [5], [8], [16], [27], [32]. Thus, we can also adapt these advanced methods to make joint optimization on traffic routing, and further design our own traffic routing method and modify the constraint of Eq. (10) if needed.

To determine whether request $r \in R$ is accepted or not, we should check that if any VNF-FP's response latency exceeds its limitation T_r. For $t \in T$, the response latency of each VNF-FP $P_r^i \in P_r$ can be represented by:

$$\tau_{P_r^i}^{rsp} = \tau_r^{cst} + \sum_{j=1}^{|P_r^i|} \sum_{n \in N} t_{r(p_j^i)}^{n} \cdot x_{n,t}^{r(p_j^i)} + \sum_{j=1}^{|P_r^i|-1} \sum_{n_1,n_2 \in N} t_{l=(n_1,n_2)}^{lnk} \cdot x_{n_1,t}^{r(p_j^i)} \cdot x_{n_2,t}^{r(p_{j+1}^i)}. \tag{11}$$

Then, when a request $r \in R$ is accepted, the maximum response latency of all its VNF-FPs should not exceed its response latency limitation T_r. Thus, $\forall t \in T$, $r \in R$, y_r can be represented as below:

$$y_r = \begin{cases} 1, \sum_{i=1}^{|r|} \sum_{n \in N} x_{n,t}^{r(i)} = |r| \text{ and } \max_{i=1}^{|P_r|} \tau_{P_r^i}^{rsp} \le T_r, \\ 0, \sum_{i=1}^{|r|} \sum_{n \in N} x_{n,t}^{r(i)} < |r| \text{ or } \max_{i=1}^{|P_r|} \tau_{P_r^i}^{rsp} > T_r. \end{cases} \tag{12}$$

Now we introduce the costs of VNF deployment and migration. First, once a request $r \in R$ is placed, it will keep consuming the demanded resources of both servers and links for τ_r^{ser} time slots if it is not migrated. As the network states and request states change, the available resource capacity $c_{n,t}$ and $b_{l,t}$ will also be updated. Note that the VNF interference ratio will also be re-calculated since the resource supply and demand are both changed. To decide whether it is worth migrating some resource-consuming VNFs to accept more arriving requests, we introduce the VNF migration cost $M_{n_1,n_2,t}^{r(i)}$ to represent the total amount of bandwidth resource (in Mbps or Gbps) it occupies to migrate VNF $r(i) \in F$ from node n_1 to node n_2 to continue processing request r's unprocessed packets. We have:

$$\begin{aligned} &\forall t \in T, r \in R, r(i) \in F, n_1, n_2 \in N, \\ &M_{n_1,n_2,t}^{r(i)} = w(n_1,n_2) \cdot x_{n_1,t-1}^{r(i)} \cdot x_{n_2,t}^{r(i)} \cdot d_{r(i)}, \end{aligned} \tag{13}$$

where $w(n_1, n_2)$ represents how much bandwidth (in Mbps or Gbps) it takes to migrate a unit amount of resource from node n_1 to node n_2, which is a distance-related parameter.

Our objective is to **maximize the total reward of accepted requests** (i.e., the total throughput minus the weighted total VNF migration cost). Since we have derived the equivalent throughput $q_{j,r}^{i*}$ along each VNF-FP $P_r^i \in P_r$, the egress throughput of VNF-FP P_r^i can be expressed as $q_{|P_r^i|-1,r}^{i*}$. Thus, if request $r \in R$ is accepted, we can represent its total throughput by summing up all its VNF-FPs' egress throughputs, i.e., $\sum_{i=1}^{|P_r|} q_{|P_r^i|-1,r}^{i*}$. Finally, the **Online Interference-aware VNF Deployment and Migration (OIVDM) problem in 5G network slice** can be formulated as follows:

$$\begin{aligned} \max \sum_{t \in T} \Big(&\sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|P_r^i|-1,r}^{i*} \cdot y_r - \\ &\beta \sum_{r(i) \in F} \sum_{n_1 \in N} \sum_{n_2 \in N} M_{n_1,n_2,t}^{r(i)} \Big), \\ s.t.\ &(1), (2), (3), (4), (5), (6), (7), (8), \\ &(9), (10), (11), (12), (13), \end{aligned} \tag{14}$$

where β is a parameter for normalization.

To deal with the real-time VNF deployment and migration problem, the NFV system needs to know the future information (i.e., the future arriving requests, their required resources and QoS requirements) so as to accept more requests with minimized migration costs. However, it is usually difficult to get the future information to make globally optimal solution in a realistic NFV system. Thus, without considering VNF migration across different time slots, we can divide the online optimization problem Eq. (14) into T one-shot throughput-maximizing problems, i.e., the **One-shot Interference-aware VNF Deployment (OIVD) problem**, which can be formulated as follows:

$$\begin{aligned} &\max \sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|P_r^i|-1,r}^{i*} \cdot y_r, \\ s.t.\ &(1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12). \end{aligned} \tag{15}$$

D. Problem Complexity

Theorem 1. *The OIVD problem defined in Eq.* (15) *is NP-hard.*

Proof: See Appendix A in online Supplementary Material.

Theorem 2. *The OIVDM problem defined in Eq.* (14) *is NP-hard.*

Proof: See Appendix B in online Supplementary Material.

IV. Algorithm Design

To solve the OIVD problem, in this section, we first introduce AIA, a one-shot adaptive interference-aware algorithm for placing a given set of requests' VNFs in a 5G network slice with maximized total throughput. Then we propose an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) to solve the OIVDM problem, which can achieve real-time VNF deployment and cost-efficient migration in a 5G network slice with maximized total reward. Afterward, we provide theoretical analyses of AIA and OLAIA in terms of optimality, competitive ratio and algorithm complexity.

A. Adaptive Interference-aware Algorithm (AIA)

Due to the NP-hardness of Eq. (15), it is computational-expensive to derive the optimal solution for a large-size network. To this end, we propose AIA, a one-shot throughput-maximizing heuristic algorithm, which is applicable to various service-customized 5G network slices. The core concept of AIA is based on the following *four-step strategy*, and the procedure of AIA is listed in Algorithm 1.

Step 1: gain the throughput of "high-yield" requests. Since there may be multiple requests arriving at a 5G network slice simultaneously, to achieve Eq. (15), we can preferentially deploy the "high-yield" request that produces the most throughput while consuming the fewest resources. Thus we denote the *yield* of each request $r \in R$ by Ψ_r,

$$\Psi_r = \frac{\sum_{i=1}^{|P_r|} q^i_{|P^i_r|-1,r}}{\sum_{i=1}^{|r|} d_{r(i)}}, \tag{16}$$

where $\sum_{i=1}^{|P_r|} q^i_{|P^i_r|-1,r}$ refers to the ideal total throughput of a request (without considering the VNF interference) and $\sum_{i=1}^{|r|} d_{r(i)}$ refers to the sum of resources it consumes. The higher Ψ_r is, the more throughput that request r produces. Thus, we sort all the requests $r \in R$ by their Ψ_r in descending order, and primarily place the one with the highest Ψ_r.

Step 2: satisfy the longest VNF-FP's response latency. Since a request $r \in R$ is accepted only if all its VNF-FPs' response latency does not exceed the response latency limitation T_r. Thus, for each request $r \in R$, it is suggested to primarily check whether its longest VNF-FP $\max r(P^i_r)$ can be satisfied. Besides, primarily placing the longest VNF-FP results in a higher probability of *reusing* its VNFs by other VNF-FPs, which can effectively reduce the number of unplaced VNFs so as to improve the VNF deployment efficiency.

Step 3: adapt to various QoS requirements. Due to the service-customized properties of 5G network slices, the requests' QoS requirements are usually at the same order of magnitude, which actually makes it convenient for service deployment. Considering the latency requirements, we first weight the latency limitation of request $r \in R$ by its logarithm $\lg(T_r)$. Consequently, a request with a lower weight $\lg(T_r)$ means it is more latency-sensitive, whose VNFs should be preferentially placed in edge cloud servers and vice versa. Assume that W denotes the latency bound, for request $r \in R$, $\lg(T_r)/W$ represents the probability of placing VNFs in the core cloud servers and $(1 - \lg(T_r)/W)$ represents the probability of placing VNFs in edge cloud servers. Note that we can also add the weight of bandwidth requirement of each request $r \in R$ by its logarithm $\lg(\lambda_r)$ and choose the link based on $\lg(\lambda_r)/W'$, where W' is the bandwidth bound for calculating weights. In particular, we use four sets to distinguish between the edge cloud servers (i.e., N_{e_1}, $N_{e_2} \subseteq N_e$) and the core cloud servers (i.e., N_{c_1}, $N_{c_2} \subseteq N_c$) which have sufficient remaining resource to place a VNF, and to further distinguish whether they have placed any VNF (i.e., N_{e_1} and N_{c_1}) or not (i.e., N_{e_2} and N_{c_2}) for considering the VNF interference. Then AIA orderly places the VNF in these four sets of servers based on the calculated weight $\lg(T_r)/W$, as listed in Line 17, Line 19, and Line 24 of Algorithm 1.

Algorithm 1 Adaptive Interference-aware Algorithm (AIA) Procedure

1: **Input**: $G = (N, L)$, R, $\forall t \in T$
2: **Output**: $x^{r(i)}_{n,t}$, y_r, ω (the total accepted throughput)
3: **Begin**: Initiate $\forall n \in N$, $\omega = 0$;
4: Sort all $r \in R$ by its Ψ_r in descending order;
5: **while** $R \neq \emptyset$ **do**
6: Get r with the maximal Ψ_r and sort all $P^i_r \in P_r$ by its $|P^i_r|$ in descending order;
7: **while** $P_r \neq \emptyset$ **do**
8: Get P^i_r with the maximal $|P^i_r|$;
9: **for** $j = 1$ to $|P^i_r|$ **do**
10: **if** $\sum_{n \in N} x^{r(p^i_j)}_{n,t} = 1$ **then**
11: $j = j + 1$, continue;
12: **else**
13: $r(j) = r(p^i_j)$;
14: **if** $N_{e_1} \cup N_{e_2} \cup N_{c_1} \cup N_{c_2} = \emptyset$ **then**
15: $y_r = 0$, $R = R - \{r\}$, deploy the next $r \in R$;
16: **else if** $j \leq \lceil (|P^i_r|(1 - \lg(T_r)/W)) \rceil$ **then**
17: Preferentially place f_j at $n \in N_{e_1}$ with the largest $c_{n,t}$, then orderly place in N_{c_1}, N_{e_2} and N_{c_2};
18: **else**
19: Preferentially place f_j at $n \in N_{c_1}$ with the largest $c_{n,t}$, then orderly place in N_{e_1}, N_{c_2} and N_{e_2};
20: **end if**
21: **if** Eq. (10) = 1 **then**
22: $x^{r(j)}_{n,t} = 1$, $j = j + 1$;
23: **else**
24: Place f_j at other server n orderly in N_{e_1}, N_{c_1}, N_{e_2} and N_{c_2} until Eq. (10)= 1, $x^{r(j)}_{n,t} = 1$, $j = j + 1$;
25: **end if**
26: **if** $j = |P^i_r|$ **then**
27: **while** $\tau^{rsp}_{P^i_r} > T_r$ **do**
28: Consolidate VNFs with the largest $q^i_{j,r}$, and get $q^{i*}_{j,r}$;
29: **if** $\tau^{rsp}_{P^i_r} \leq T_r$ **then**
30: $\omega = \omega + q^{i*}_{|P^i_r|-1,r}$,
31: $P_r = P_r - \{P^i_r\}$, deploy VNF-FP P^{i+1}_r;
32: **end if**
33: **end while**
34: $y_r = 1$, $R = R - \{r\}$, and deploy the next $r \in R$;
35: **end if**
36: **end if**
37: **end for**
38: **end while**
39: **end while**

Step 4: handle the VNF interference. Last but not least, note that VNF consolidation will cause severe throughput degradation. Thus, to maximize the total accepted throughput, we primarily consider not consolidating VNFs in the same server. However, when doing this results in the violations of constraint Eq. (7), Eq. (10) and/or Eq. (12), we consider consolidating two VNFs (e.g., $r(i) \in F_r$ and $r(i+1) \in F_r$) of request $r \in R$ if the throughput between them is the largest (i.e., the largest $a_{r(i,i+1)} \in A_r$). This way, we reserve as many bandwidth resources as possible for deploying other VNF-FPs. Note that if consolidating two VNFs still violates the constraints (e.g., Eq. (12)), we gradually increase the number of consolidated VNFs until all the constraints are satisfied.

We use ω to represent the total accepted throughput as defined in Eq. (15). Note that *Step 1* (Line 4-6) and *Step 2* (Line 7-9) are complementary to each other to improve the total throughput of accepted requests. Line 12 judges whether VNF $r(p^i_j) \in F$ has been placed in other VNF-FPs. Line 14-15 means no server has sufficient resources for placing f_j, then request r is rejected, and AIA moves on to the next request. Note that Line 17, Line 19, and Line 24 choose the server

with the largest available resource $c_{n,t}$. This way, on the one hand, it increases the demand-supply ratio $\alpha_{n,f,t}$ as defined in Eq. (2); on the other hand, it reserves more resources for deploying other VNFs. When VNFs are consolidated in *Step 4* (Line 27-33), we revise the actual throughput along each VNF-FP by Eq. (8).

B. Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA)

In the procedure of AIA, if there are not enough remaining resources or the QoS requirements can not be satisfied, some arriving requests will be rejected. However, to further improve the long-term total reward in a 5G network slice, one possible solution is worth considering: *can we migrate some deployed VNFs to release resources and "make place" for more new arriving requests to be accepted?*

An aggressive solution is to migrate VNF(s) whenever an arriving request's VNFs can not be totally deployed. However, the cost of frequent VNF migration will be much higher than the revenue of accepting requests, which makes this solution not practical. Nevertheless, if we never migrate some long-term resource-consuming VNFs, more arriving requests might be rejected due to the resource shortage. Thus, the essential questions are to decide *when to migrate a VNF, which VNF should be migrated and migrated to which server?*

We design an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) for VNF deployment and migration, which is inspired by the work [49]. The key idea is to tolerate as much non-migration cost as possible until it significantly exceeds the migration cost, where the non-migration M_t^{non} at time slot $t \in T$ is defined by the total throughput of rejected requests as follows:

$$\forall t \in T, M_t^{non} = \sum_{r \in R_t} (1 - y_r) \cdot \lambda_r, \tag{17}$$

where $R_t = \{r \in R | \tau_r^{ari} = t\}$ is the set of all arriving requests at time slot $t \in T$. To decide when to migrate a VNF $\hat{f} \in F$, we define a judgment condition to check whether the accumulated non-migration cost is at least η times of the migration cost, i.e., $M_{n_1,n_2,t}^{\hat{f}} \leq \frac{1}{\eta} \sum_{t^*=\hat{t}}^{t} M_{t^*}^{non}$, where $\hat{t}$ is the last time slot when a VNF is migrated. Note that η is used as an indicator to control the VNF migration frequency. Thus, a larger η means to tolerate more non-migration cost (i.e., more lost throughput of rejected requests) in OLAIA. We can vary the value of η for different 5G network slices to satisfy their specific QoS requirements. We define a set R_t^{rej} to include all the rejected requests before time slot $t \in T$, and a set R_t^{ser} to include all the requests in service at time slot $t \in T$.

Next, to decide which VNF $\hat{f} \in F$ should be migrated, we should also jointly consider its resource consumption $d_{\hat{f}}$ and also its location in VNF-FG. For the first thing, $d_{\hat{f}}$ should consume as few resources as possible to minimize the migration cost; for the second thing, $\hat{f} \in F$ should better be the VNF which has *the minimum effect* on the whole VNF-FG. Thus, migrating the last VNF in a VNF-FP which consumes the fewest resources is a good choice in most cases. For example, as plotted in Fig. 2, VNF6 is the targeted VNF for migration if it consumes the fewest resources, since it only affects the first VNF-FP (i.e., $P_r^1 = (1, 2, 4, 6)$). The second choice is to migrate VNF7 if it consumes fewer resources than VNF6, since it affects the other two VNF-FPs (i.e., $P_r^2 = (1, 2, 4, 5, 7)$ and $P_r^3 = (1, 3, 5, 7)$). Note that we take the resource consumption as the primary factor to determine the target VNF for migration, since it directly affects the migration cost as defined in Eq. (13). In particular, we use a function $\hat{f} = find(R_t^{ser})$ to determine a target VNF $\hat{f} \in F$ for migration, whose input is the set of requests in service R_t^{ser}. $find(R_t^{ser})$ first finds the last VNF in each VNF-FP of R_t^{ser}, then it compares their resource consumption and preferentially outputs the target VNF $\hat{f}$ that consumes the fewest resources.

Finally, we decide where the target VNF $\hat{f} \in F$ should be migrated to. When a request $r \in R$ is rejected, mostly it is due to the resource shortage on edge cloud servers for placing its required VNFs, while placing them on core cloud servers will exceed its response latency limitation. Thus, in order to accept new arriving requests, our idea is to migrate the target VNF $\hat{f}$ from an edge cloud server $n_1 \in N_e$ that satisfies $x_{n_1,t}^{\hat{f}} = 1$, to a close core cloud server $n_2 \in N_c$ that has enough resources for placing $\hat{f}$ (i.e., $c_{n_2,t} \geq d_{\hat{f}}$) with minimized $w(n_1, n_2)$. The new response latency of VNF $\hat{f}$'s request $\hat{r} \in R$ should still not exceed its response latency limitation $T_{\hat{r}}$ after migration, while the service time $\tau_{\hat{r}}^{ser}$ of $\hat{r}$ has not terminated. As illustrated, we also define a function $n_2 = migr(\hat{f}, n_1)$, whose input $n_1 \in N_e$ satisfies $x_{n_1,t}^{\hat{f}} = 1$ at time slot $t \in T$. In particular, $migr(\hat{f}, n_1)$ first finds a set

Algorithm 2 Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) Procedure

1: **Input**: $G = (N, L)$, R, T
2: **Output**: $x_{n,t}^{r(i)}$, y_r, Ω (the total reward)
3: **Begin**: Initiate $\forall n \in N$, $\Omega = 0$, $t = 1$, $\hat{t} = 1$, $\hat{f} = 0$, $R_t = \emptyset$, $R_t^{rej} = \emptyset$,$R_t^{ser} = \emptyset$;
4: **while** $t < T$ **do**
5: **for** $R_t^{ser} \neq \emptyset$ and $r \in R_t^{ser}$ **do**
6: **if** $t > \tau_r^{ari} + \tau_r^{ser}$ **then**
7: $s_{r,t} = 0$, $R_t^{ser} = R_t^{ser} - r$;
8: **end if**
9: **end for**
10: $R_t = \{r \in R | \tau_r^{ari} = t\}$;
11: Call AIA(G, R_t, t);
12: Update G, $x_{n,t}^{r(i)}$, y_r, $s_{r,t}$, and update Ω by $\Omega = \Omega + \omega$;
13: $R_t^{rej} = R_t^{rej} \cup \{r \in R_t | y_r = 0\}$, $R_t^{ser} = R_t^{ser} \cup \{r \in R_t | s_{r,t} = 1\}$;
14: $\hat{f} = find(f)$, $n_2 = migr(\hat{f}, n_1)$;
15: Compute M_t^{non} and $M_{n_1,n_2,t}^{\hat{f}}$ by Eq. (17) and Eq, (13), and update ϵ by calculating $\max_{t \in [1,T]} M_t^{non}$ and $\min_{t \in [1,T]} M_t^{non}$;
16: **if** $M_{n_1,n_2,t}^{\hat{f}} \leq \frac{1}{\eta} \sum_{t^*=\hat{t}}^{t} M_{t^*}^{non}$ **then**
17: **if** $n_1 \neq n_2$ **then**
18: Build a service instance of $\hat{f}$ on node n_2;
19: Steer the unprocessed packets of $\hat{f}$ from n_1 to n_2;
20: Shut down the service instance of $\hat{f}$ on node n_1 and release its occupied resources;
21: Update G, $x_{n,t}^{r(i)}$, and update Ω by Eq. (14);
22: Call AIA(G, R_t^{rej}, t);
23: Update G, $x_{n,t}^{r(i)}$, y_r, $s_{r,t}$, and update Ω by $\Omega = \Omega + \omega$;
24: $R_t^{ser} = R_t^{ser} \cup \{r \in R_t^{rej} | s_{r,t} = 1\}$, $R_t^{rej} = R_t^{rej} - \{r \in R_t^{rej} | y_r = 1\}$, $\hat{t} = t$;
25: **end if**
26: **end if**
27: $t = t + 1$, $R_t = \emptyset$;
28: **end while**

$N^* = \{n \in N_c | c_{n,t} \geq d_{\hat{f}}\}$ and outputs $n_2 \in N^*$ whose $w(n_1, n_2)$ is minimal, which means n_2 is the closest node to n_1 for migrating $\hat{f}$ with the least migration cost. Note that if $N^* = \emptyset$, then redefine $N^* = \{n \in N_e | c_{n,t} \geq d_{\hat{f}}\}$. Otherwise, $n_2 = 0$, which means there is no available node in both core cloud servers and edge cloud servers for migrating $\hat{f}$, thus $M^{\hat{f}}_{n_1,n_2,t} = +\infty$. Note that migrating $\hat{f}$ from $n_1 \in N_e$ to $n_2 \in N_e$, from $n_1 \in N_c$ to $n_2 \in N_c$ and even from $n_1 \in N_c$ to $n_2 \in N_e$ are also acceptable if the total reward can be further improved.

The whole procedure of OLAIA is listed in Algorithm 2. In a time slot $t \in T$, OLAIA first updates the states of all requests in R^{ser}_t and updates R^{ser}_t (Line 5-9). Then OLAIA tries to place all the arriving requests in R_t by calling AIA (Line 10-11). Next, OLAIA updates the network state, the request state, the total reward Ω, R^{rej}_t, R^{ser}_t, etc., (Line 12-13). Then, OLAIA finds the target VNF for migration with its target server, calculates M^{non}_t and $M^{\hat{f}}_{n_1,n_2,t}$, and updates ϵ, defined by $\epsilon = \max_{t\in[1,T]} \frac{\max_{t\in[1,T]} M^{non}_t}{\min_{t\in[1,T]} M^{non}_t}$ (Line 14-15). OLAIA checks whether the judgment condition of VNF migration is satisfied (Line 16). If so, OLAIA migrates $\hat{f}$ from n_1 to n_2 to process its unprocessed packets, releases its previously occupied resources, and updates Ω by subtracting the weighted migration cost as defined in Eq. (14) (Line 17-21). Then OLAIA calls AIA again and tries to place the previously rejected requests in R^{rej}_t (Line 22). Then all the states, sets and variables are updated (Line 23-24). Finally, the time slot t moves on and R_t is reset (Line 27).

C. Theoretical Analysis

We first give a brief optimality analysis of AIA.

Theorem 3. *The worst-case performance bound of AIA is* $Z \cdot \lg(T_{max})/W$, *where* $T_{max} = \max_{r\in R} T_r$ *and* $Z = \min_{r\in R, n\in N} \prod_{i=1}^{\lfloor(|r|-1)/2\rfloor} \alpha_{n,r(i),t} \in (0, 1]$.

Proof: See Appendix C in online Supplementary Material.

Note that this worst-case performance bound indicates the largest performance gap between AIA and the optimal one-shot throughput-maximizing solution in achieving the total accepted throughput for the same set of requests as input.

Then we derive the competitive ratio of OLAIA.

Lemma 1. *The overall migration cost in* $[1, T]$ *is at most* $1/\eta$ *of the overall non-migration cost in this period, i.e.,* $\sum_{t=1}^{T} M^{\hat{f}}_{n_1,n_2,t} \leq \frac{1}{\eta} \sum_{t=1}^{T} M^{non}_t$.

Proof: Let $\hat{t}_i$ be the time slot for the i-th occurrence of a VNF migration. Within the period $[\hat{t}_i, \hat{t}_{i+1} - 1]$, only one migration occurs at $\hat{t}_i$ when the judge condition is satisfied. Thus, $\forall i \in \mathbb{N}$, $\hat{t}_{i+1} - 1 < T$, we can conclude that the non-migration cost is at least is η times of the migration cost in $[\hat{t}_i, \hat{t}_{i+1} - 1]$. Thus, we have:

$$\sum_{t=1}^{T} M^{\hat{f}}_{n_1,n_2,t} \leq \frac{1}{\eta} \sum_{t=1}^{T-1} M^{non}_t \leq \frac{1}{\eta} \sum_{t=1}^{T} M^{non}_t. \tag{18}$$

■

Lemma 2. *The overall non-migration cost in* $[1, T]$ *is at most* ϵ *times of the total offline-optimal cost, i.e.,* $\sum_{t=1}^{T} M^{non}_t \leq \epsilon \sum_{t=1}^{T} M^*_t$, *where* $\sum_{t=1}^{T} M^*_t$ *is the optimal cost of the total cost, i.e.,* $\sum_{t=1}^{T} M_t = \sum_{t=1}^{T} (M^{\hat{f}}_{n_1,n_2,t} + M^{non}_t)$ *and* $\epsilon = \max_{t\in[1,T]} \frac{\max_{t\in[1,T]} M^{non}_t}{\min_{t\in[1,T]} M^{non}_t}$.

Proof: Let $M^{non^*}_t$ be the optimal non-migration cost at time slot $t \in T$, and $\min_{t\in[1,T]} M^{non}_t$, $\max_{t\in[1,T]} M^{non}_t$, $\min_{t\in[1,T]} M^{non^*}_t$ and $\max_{t\in[1,T]} M^{non^*}_t$ is the minimal-actual, maximal-actual, minimal-optimal and maximal-optimal non-migration cost in time slots $[1, T]$, respectively. Thus we have $\min_{t\in[1,T]} M^{non}_t \leq M^{non}_t \leq \max_{t\in[1,T]} M^{non}_t$ and $\min_{t\in[1,T]} M^{non^*}_t \leq M^{non^*}_t \leq \max_{t\in[1,T]} M^{non^*}_t$. By defining ϵ, which is the maximum ratio of the maximum/minimum non-migration cost incurred in time slot $[1, T]$, we have $M^{non}_t \leq M^{non}_t \cdot \frac{M^{non^*}_t}{\min_{t\in[1,T]} M^{non^*}_t} \leq \max_{t\in[1,T]} M^{non}_t \cdot \frac{M^{non^*}_t}{\min_{t\in[1,T]} M^{non}_t} \leq \epsilon M^{non^*}_t$. We have:

$$\begin{aligned} &\sum_{t=1}^{T} M^{non}_t \leq \epsilon \sum_{t=1}^{T} M^{non^*}_t \\ &\leq \epsilon \sum_{t=1}^{T} (M^{non^*}_t + M^{\hat{f}^*}_{n_1,n_2,t}) \leq \epsilon \sum_{t=1}^{T} M^*_t. \end{aligned} \tag{19}$$

■

Theorem 4. *OLAIA has a competitive ratio of* $\epsilon(1 + \frac{1}{\eta})$ *compared to the offline optimum.*

Proof: By applying Lemma 1 and Lemma 2 to the overall cost, we have:

$$\begin{aligned} &\sum_{t=1}^{T} M_t = \sum_{t=1}^{T} (M^{\hat{f}}_{n_1,n_2,t} + M^{non}_t) \\ &\leq \sum_{t=1}^{T} ((\frac{1}{\eta} + 1) M^{non}_t) \leq \epsilon(1 + \frac{1}{\eta}) \sum_{t=1}^{T} M^*_t. \end{aligned} \tag{20}$$

Therefore, we prove that the OLAIA can gain a competitive ratio of $\epsilon(1 + \frac{1}{\eta})$ compared to the offline optimum, which can represent the worst-case performance bound of the algorithm. And the value of ϵ is mainly affected by the resource demand of requests, which varies in different 5G network slices. ■

The competitive ratio reveals the least gained total reward over the global offline optimum with appropriate VNF migration and redeployment of previously rejected requests. In our experiments, to achieve the maximal total reward, η is set up to 1.3 and ϵ is given by 1.6, and the competitive ratio is as low as 2.83. Evaluation results in Sec. VI reveals the superior performance of OLAIA in practice.

D. Complexity Analysis

First, for AIA, sorting R (Line 4) requires $O(|R| \log_2 |R|)$ computation and sorting P_r (Line 6) requires $O(|P_r| \log_2 |P_r|)$ computation. Next, the first **while**-loop (Line 5) terminates in $\mathbf{R} = |R|$ iterations and the second **while**-loop (Line 7) terminates in $\mathbf{L} = |P_r|$ iterations. The **for**-loop (Line 9) terminates in $\mathbf{M} = \max_{i=1} |P^i_r|$ iterations and the last **while**-loop (Line 27) also terminates in $\mathbf{M}$ iterations. Thus, the

computation complexity of AIA is $O(\mathbf{R}\log_2\mathbf{R}+\mathbf{R}(\mathbf{L}\log_2\mathbf{L}+\mathbf{LM^2}))=O(\mathbf{R}(\log_2\mathbf{R}+\mathbf{L}\log_2\mathbf{L}+\mathbf{LM^2}))$.

For OLAIA, there is one **while**-loop (Line 4), and within it, OLAIA calls AIA twice and has a **for**-loop (Line 5), which can be done in $O(\mathbf{R})$ iterations. Note that function $find(f)$ can be done by sorting all the placed VNFs in $O(\mathbf{RLM})$ iterations and function $migr(\hat{f}, n_1)$ requires $O(\mathbf{N^2})$ computations, where $\mathbf{N} = |N_e| + |N_c|$. Thus, the computation complexity of OLAIA is $O(\mathbf{T}(\mathbf{R}+O(AIA)+\mathbf{RLM}+\mathbf{N^2}) = O(\mathbf{T}(\mathbf{R}\log_2\mathbf{R}+\mathbf{RL}\log_2\mathbf{L}+\mathbf{RLM^2}+\mathbf{N^2}))$.

V. Performance Evaluation

To demonstrate the effectiveness of AIA and OLAIA, we simulate two typical 5G network slices, i.e., **CASE I: an autonomous driving network slice** and **CASE II: a 4K/8K HD video network slice**. We first introduce the evaluation setup and then discuss the performance evaluation results.

A. Evaluation Setup

5G network slice topology. We use the GT-ITM[3] tool to generate the 5G network slice topology. In each 5G network slice, we divide the edge/core cloud servers by their distances from the end-users as discussed in [39], and we scale the number of edge cloud servers from 5 to 300 and double the number of core cloud servers scaling from 10 to 600. According to [28], we configure the edge cloud servers with 10∼40 units of CPU resource and 1∼16 units of memory resource, while core cloud servers with 20∼200 and 16∼64 units. The CPU and memory resources required by each VNF are randomly distributed in 1∼20 units and 1∼4 units.

Parameter of 5G use cases. As listed in Table. III, we set the values of parameters based on [9], [38], [50]. The VNF-FGs are generated in the same way as stated in [26], including linear chains, split-and-merged graphs, and other complex one-ingress multi-egress directed graphs. In both 5G use cases, the number of requests we simulate ranges from 5 to 1,000 for AIA. To evaluate the real-time performance of OLAIA, we use the real-world traces from Alibaba [51], which includes a sum of 30,000 requests arriving in 6,000 time slots with request arriving time, service time, network traffic, CPU and memory utilization, etc. We set the length of a time slot by 10 ms for the autonomous driving case and 500 ms for the 4K/8K HD video case.

B. Performance Evaluation Results of AIA

To evaluate the performance of AIA, we compare it with the following two state-of-the-art heuristic algorithms [52]:

- **Shortest Path Heuristic (SPH)**: it processes the requests based on a first-come-first-served basis and deploys each VNF-FP of a request along the shortest path between the ingress point and the egress point(s).
- **Greedy on Used Server (GUS)**: it aims to minimize the number of occupied servers so as to reduce the OPEX and energy consumption, which preferentially places VNFs in the used servers to avoid occupying extra unused servers.

[3]GT-ITM: https://www.cc.gatech.edu/projects/gtitm/

TABLE III: Key Parameters & Values.

Parameter	CASE I: Autonomous Driving	CASE II: 4K/8K HD Video
Number of VNFs	1∼4	1∼7
Service-required VNFs*	NAT, FW, TM, ADNF, etc.	NAT, FW, TM, VOC, IDCS, CNF, DNF, etc.
$\lvert P_r\rvert$	1∼3	1∼6
t_f^n	0.1 ms	1 ms
t_l^{lnk}	0.01∼0.5 ms	1∼5 ms
$b_{l,t}$	100 Mbps∼1 Gbps	1 Gbps∼20 Gbps
λ_r	1∼10 Mbps	0.2∼4 Gbps
T_r	1 ms	100 ms

*NAT: Network Address Translation, FW: Firewall, TM: Traffic Monitor, ADNF: Autonomous Driving required Network Function, VOC: Video Optimization Controller, CNF: Compression Network Function, DNF: Decompression Network Function

Execution time. Since AIA's computation complexity is positively correlated with the number of servers and requests, we scale the number of servers from 15 to 900 and requests from 5 to 1,000 to demonstrate the efficiency of AIA. Fig. 4 depicts the execution time of AIA with the increasing number of total servers. As plotted, it takes less than 100 milliseconds to get a solution for a small-scale network with less than 150 servers. When the number of servers increases, the execution time of AIA also increases, but it is still acceptable (e.g., 0.89 seconds for a network with 300 servers). Fig. 5 shows that AIA can converge in 1 second for handling less than 200 requests. In short, AIA can efficiently get a feasible solution even with large amounts of requests in a 1,000-host network.

Total throughput of accepted requests. This is the primary performance metric as defined in Eq. (15). Fig. 6 and Fig. 7 plot the average throughput in CASE I and CASE II, respectively, as the number of requests scales from 5 to 1,000. As plotted, AIA achieves averagely 10.01% and 15.02% more throughput of accepted requests than SPH and GUS in CASE I, and correspondingly 25.99% and 30.52% more throughput in CASE II. Notably, AIA improves the average throughput by 22.67% and 43.54% when dealing with 1,000 latency-sensitive requests in CASE I as compared with SPH and GUS. This benefits from the *four-step strategy* of AIA, especially from *Step 1* and *Step 4*, where AIA reduces the VNF interference to improve the achieved throughput by decreasing the probability of VNF consolidation.

Request acceptance ratio. Next, we compare the request acceptance ratio in both cases, which are plotted in Fig. 8 and Fig. 9. In CASE I, the request acceptance ratio decreases notably as the number of requests increases, i.e., from 100% to 53.94%. When the number of requests reaches 200, more than 30% of requests can not be deployed, mostly due to the violations on response latency constraint. However, in CASE II, the request acceptance ratio decreases slightly as the number of requests increases, i.e., from 100% to 94.04%. This is because the response latency requirement is not as strict as CASE I. The average request acceptance ratio of AIA, SPH, GUS is 79.02%, 81.50%, 76.90% in CASE I and 96.07%, 97.83%, 95.12% in CASE II. Note that even though SPH's request acceptance ratio is about 2% higher than AIA, its total throughput is averagely 18% less than AIA. This is because SPH prefers consolidating as many VNFs as possible in edge cloud servers to reduce response latency

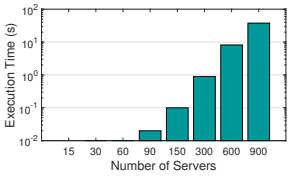

Fig. 4: The execution time with different number of servers.

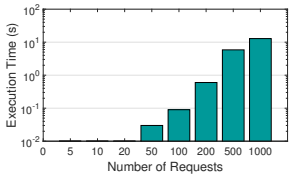

Fig. 5: The execution time with different number of requests.

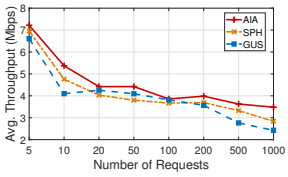

Fig. 6: The average total throughput in CASE I.

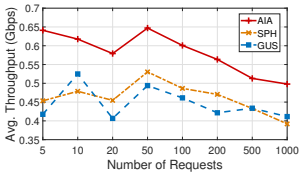

Fig. 7: The average total throughput in CASE II.

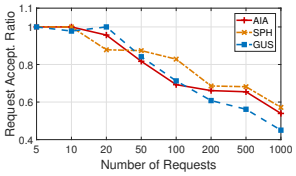

Fig. 8: The request acceptance ratio in CASE I.

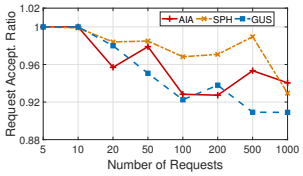

Fig. 9: The request acceptance ratio in CASE II.

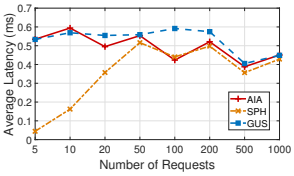

Fig. 10: The average response latency in CASE I.

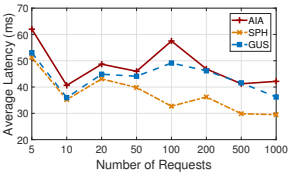

Fig. 11: The average response latency in CASE II.

while ignores the VNF interference and causes degradation in the total throughput. Besides, unlike *Step 1* in AIA, SPH merely processes requests based on a first-come-first-served basis, thus SPH may lose some "high-yield" (as defined in Eq. (16)) requests and achieves lower throughput.

Average response latency. Finally, we also compare the average response latency of accepted requests. As plotted in Fig. 10 and Fig. 11, the average response latency fluctuates around 0.4∼0.6 ms in the autonomous driving case and around 30∼50 ms in the 4K/8K HD video case. Note that SPH shows a good advantage in reducing the response latency because SPH consolidates as many VNFs as possible in the edge cloud servers to reduce end-to-end response latency. However, this can inevitably cause throughput degradation, as shown in Fig. 6 and Fig. 7. Besides, AIA can also meet the latency requirements in both 5G use cases.

C. Performance Evaluation Results of OLAIA

To further evaluate OLAIA, we first adjust the frequency of VNF migration η to see how OLAIA performs. Then, we compare OLAIA with the following four schemes:

- **Online Non-migration with AIA (ON+A)**: it merely places VNFs in each time slot by AIA without migration, which can be a baseline for comparison.
- **Online Throughput-maximizing with AIA (OT+A)**: it optimally maximizes the total throughput of all arriving requests without considering the migration cost, which usually results in frequent migration of VNFs.
- **Online Lazy-migration with Greedy on core cloud server (OL+G)**: it combines the online lazy-migration strategy with the greedy-based VNF placement approach, which preferentially places VNFs in the core cloud servers to reserve resources of edge cloud servers and also reduce migration cost.
- **Online Lazy-migration with Shortest path heuristic (OL+S)**: it combines the online lazy-migration strategy with SPH [52], which deploys each VNF-FP of a request along the shortest path.

Effect of the migration frequency. As defined previously, we use η to control the frequency of VNF migration. Fig. 12 and Fig. 13 plot the results of migration time and total reward in 100 time slots in CASE I with different values of η, i.e., η = 0.3, 1.0, 1.7, and 10.0. Consequently, with larger η, the migration time reduces dramatically since OLAIA can tolerate more non-migration cost. Note that when η reaches 10.0, the migration time is reduced to 0. This is because the migration cost is much higher than the weighted accumulated non-migration cost in 100 time slots. In other words, the judgment condition is always false with $\eta = 10.0$ and t moves to 100 time slots, and thus no VNF is migrated. However, the total reward first increases and then drops after reaching a certain value. Thus, there must exist an optimal value of η to maximize the total reward. For CASE I, the requests are more sensitive to latency, where a smaller η is better for improving the total reward; for CASE II, the migration cost is much higher due to the large amount of data to transmit, thus reducing the migration frequency with a larger η is better.

Average total reward. The average total reward refers to the total throughput of accepted requests minus the total cost, which is the primary performance metric defined in Eq. (14). Fig. 14 and Fig. 15 show the results of the average total reward in every 1,000 time slots for CASE I and CASE II, respectively. As plotted, OLAIA can always maximize the total reward, with η = 1.0 and η = 1.3 for the two evaluated scenarios. The average total reward is 39.61, 34.45, 32.42, 33.03, and 11.19 Mbps for OLAIA, OT+A, OL+G, OL+S, and ON+A in CASE I, and 7.54, 4.99, 6.72, 6.01, and 4.48 Gbps for the corresponding five schemes in CASE II. Notably, OLAIA achieves about 3.54× and 1.68× total reward than the baseline ON+A algorithm, which verifies the benefits of VNF migration with an appropriate migration frequency.

Average total throughput. We also show the average total throughput of accepted requests in every 1,000 time slots in Fig. 16 and Fig. 17 for CASE I and CASE II, respectively. As we can see, OT+A achieves the highest throughput with frequent VNF migrations; the following are OLAIA, OL+S, OL+G, and the baseline ON+A achieves the lowest throughput. The average throughput is 59.84, 65.36, 45.98, 52.01, and 11.21 Mbps for OLAIA, OT+A, OL+G, OL+S, and ON+A in CASE I, and 12.75, 14.03, 10.77, 10.92, and 4.55 Gbps for the corresponding five schemes in CASE II. Even though OT+A achieves about 9.22% to 10.04% more throughput than OLAIA, its migration cost is nearly two times of OLAIA.

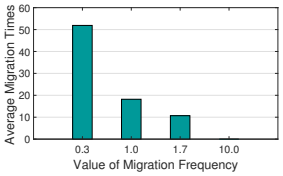

Fig. 12: The average migration times with different migration frequency η in CASE I.

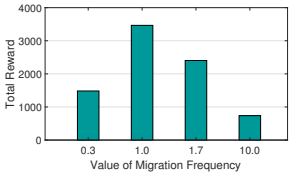

Fig. 13: The total reward with different migration frequency η in CASE I.

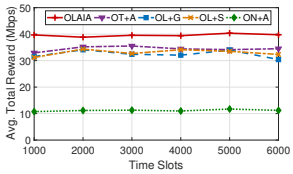

Fig. 14: The average total reward in CASE I.

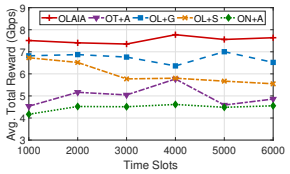

Fig. 15: The average total reward in CASE II.

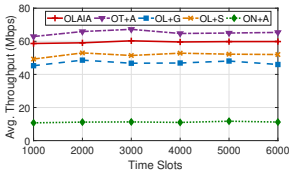

Fig. 16: The average throughput in CASE I.

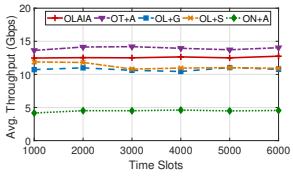

Fig. 17: The average throughput in CASE II.

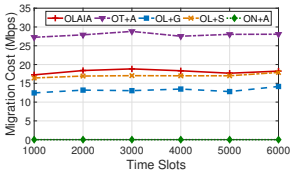

Fig. 18: The migration cost in CASE I.

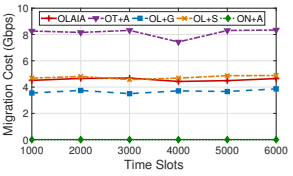

Fig. 19: The migration cost in CASE II.

Migration cost. Finally, we give a cost-analysis to highlight the superiority of OLAIA. We show the results of the migration cost in every 1,000 time slots in Fig. 18 and Fig. 19 for CASE I and CASE II, respectively. As plotted, since ON+A never migrates VNFs, the migration cost is always 0. The average migration cost is 18.25, 28.07, 14.16, and 17.89 Mbps for OLAIA, OT+A, OL+G, and OL+S in CASE I, and 4.65, 8.34, 3.86, and 4.88 Gbps for these four schemes in CASE II. Note that the migration cost of OT+A is about 1.54× and 1.79× of OLAIA in the two 5G use cases, but OT+A only improves the accepted throughput by 9.16% and 10.64% as compared by OLAIA. In conclusion, OLAIA can maximize the total reward with a relatively low migration cost.

VI. Conclusion and Future Work

This paper presents the study on online VNF deployment and migration for 5G network slice. We propose a time-slot based 5G network slice model, which includes both edge cloud servers and core cloud servers, real-time network state, resource utilization and arriving requests, and we use a demand-supply model to quantify the ubiquitous VNF interference. Due to the NP-hardness of the formulated problem, we propose an Online Lazy-migration Adaptive Interference-aware Algorithm (OLAIA) for real-time VNF deployment and cost-efficient VNF migration to maximize the total reward in a 5G network slice, where the Adaptive Interference-aware Algorithm (AIA) is proposed as OLAIA's core function for placing a given set of requests' VNFs with maximized total throughput. We give theoretical analyses of both AIA and OLAIA in terms of optimality, competitive ratio, and algorithm complexity. Through trace-driven evaluations on two simulated 5G network slices, we demonstrate that AIA can improve the average throughput of accepted requests by 15.02% and 30.52% in the autonomous driving scenario and the 4K/8K HD video scenario; while OLAIA can achieve 3.54× and 1.68× the total reward than the offline non-migration algorithm AIA, and improve it by 22.18% and 51.10% in these two 5G use cases as compared with other state-of-the-art schemes.

Future work includes joint optimization on minimizing the end-to-end latency, considering the performance degradation on both latency and throughput caused by the VNF interference, design of an online fault-tolerant VNF deployment and recovery scheme, and evaluations on more 5G use cases.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] B. Chatras, U. S. T. Kwong, and N. Bihannic, "NFV enabling network slicing for 5G," in *Innovations in Clouds, Internet and Networks*, 2017.

[3] 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 Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2017.

[4] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[5] H. Yao, C. Fang, Y. Guo, and C. Zhao, "An optimal routing algorithm in service customized 5G networks," *Mobile Information Systems*, 2016.

[6] Z. Xu, F. Liu, T. Wang, and H. Xu, "Demystifying the energy efficiency of network function virtualization," in *IEEE/ACM IWQoS*, 2016, pp. 1–10.

[7] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[8] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 486–494.

[9] N. Alliance, "5G white paper," *Next generation mobile networks, white paper*, pp. 1–125, 2015.

[10] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[11] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "VNF-FG design and VNF placement for 5G mobile networks," *Science China Information Sciences*, vol. 60, no. 4, p. 040302, 2017.

[12] R. Solozabal, B. Blanco, J. O. Fajardo, I. Taboada, and J. G. Lloreda, "Design of virtual infrastructure manager with novel VNF placement features for edge clouds in 5G," in *International Conference on Engineering Applications of Neural Networks*, 2017, pp. 669–679.

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

[14] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware VNF placement and chaining based on a flexible resource allocation approach," in *International Conference on Network and Service Management*, 2018, pp. 1–7.

[15] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF placement and resource allocation for the support of vertical services in 5G networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 27, no. 1, pp. 433–446, 2019.

[16] O. Alhussein, P. T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "Joint VNF placement and multicast traffic routing in 5G core networks," in *IEEE GLOBECOM*, 2018, pp. 1–6.
[17] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "Netbricks: Taking the V out of NFV." in *USENIX OSDI*, 2016, pp. 203–216.
[18] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the performance interference of co-located virtual network functions," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 765–773.
[19] A. Tootoonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, and S. Shenker, "ResQ: Enabling SLOs in network function virtualization," in *USENIX NSDI*, 2018.
[20] X. Li, X. Wang, F. Liu, and H. Xu, "DHL: Enabling flexible software network functions with FPGA acceleration," in *IEEE ICDCS*, 2018, pp. 1–11.
[21] Q. Zhang, Y. Xiao, F. Liu, J. C. S. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *IEEE ICDCS*, 2017, pp. 731–741.
[22] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516301989
[23] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iaware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Transactions on Computers (TC)*, vol. 63, no. 12, pp. 3012–3025, 2014.
[24] M. Shifrin, E. Biton, and O. Gurewitz, "Optimal control of VNF deployment and scheduling," in *IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, 2016, pp. 1–5.
[25] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *International Conference on Network and Service Management*, 2014, pp. 418–423.
[26] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *IEEE International Conference on Cloud Computing*, 2017, pp. 376–383.
[27] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *IEEE International Conference on Cloud Networking (CloudNet)*, 2015, pp. 171–177.
[28] S. Khebbache, M. Hadji, and D. Zeghlache, "Scalable and cost-efficient algorithms for VNF chaining and placement problem," in *Innovations in Clouds, Internet and Networks*, 2017.
[29] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *IEEE GLOBECOM*, 2017.
[30] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *Journal of Lightwave Technology*, vol. 34, no. 11, pp. 2590–2600, 2016.
[31] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2016, pp. 1–9.
[32] 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 (TON)*, vol. 25, no. 4, pp. 2008–2025, 2017.
[33] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2017.
[34] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "NFVdeep: adaptive online service function chain deployment with deep reinforcement learning," in *IEEE/ACM IWQoS*, 2019, pp. 1–10.
[35] Q. Xu, D. Gao, and T. Li, "Low latency security function chain embedding across multiple domains," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.
[36] M. Li, Q. Zhang, and F. Liu, "Finedge: A dynamic cost-efficient edge resource management platform for NFV network," in *IEEE/ACM IWQoS*, 2020, pp. 1–10.
[37] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2020.
[38] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic realistic edge cloud environment," in *IEEE GLOBECOM*, 2017, pp. 1–6.
[39] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 693–701.
[40] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *IEEE GLOBECOM*, 2017, pp. 1–7.
[41] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2019, pp. 2449–2457.
[42] T. Wang, H. Xu, and F. Liu, "Multi-resource load balancing for virtual network functions," in *IEEE ICDCS*, 2017, pp. 1322–1332.
[43] X. Fei, F. Liu, H. Xu, and H. Jin, "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure," in *IEEE/ACM IWQoS*, 2017, pp. 1–10.
[44] X. Fei, F. Liu, H. Jin, and B. Li, "FlexNFV: Flexible network service chaining with dynamic scaling," *IEEE Network*, vol. 34, no. 4, pp. 203–209, 2020.
[45] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Transactions on Computers (TC)*, vol. 65, no. 8, pp. 2470–2483, 2016.
[46] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.
[47] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2017.
[48] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 51–62.
[49] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.
[50] S. B. Mer, "Advanced autonomous vehicle with 5G technology," *Int. J. Eng. Develop. Res.*, vol. 3, no. 2, pp. 2321–9939, 2015.
[51] "ClusterData2018 in Alibaba," https://github.com/alibaba/clusterdata, [Online accessed 21-April-2020].
[52] 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 (TON)*, vol. 26, no. 4, pp. 1562–1576, 2018.

Qixia Zhang received his B.Eng. degree from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2016. He is currently a Ph.D. student in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include network function virtualization, cloud computing and edge computing, datacenter and green computing, 5G network and network slicing. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019.

Fangming Liu (S'08, M'11, SM'16) received the B.Eng. degree from the Tsinghua University, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, as well as the Second Class Prize of National Natural Science Award and the First Class Prize of Natural Science of Ministry of Education in China.

Chaobing Zeng received her B.Eng. degree in the College of Computer Science and Electronic Engineering, Hunan University, China, and the M.Eng. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, China, in 2019. Her research interests include data center networking and network function virtualization.

Online Adaptive Interference-aware VNF Deployment and Migration for 5G Network Slice —Supplementary Material

Qixia Zhang, Fangming Liu*, *Senior Member, IEEE,* and Chaobing Zeng

APPENDIX A

Theorem 1. *The One-shot Interference-aware VNF Deployment (OIVD) problem defined in Eq. (15) is NP-hard.*

Proof: We construct a polynomial-time reduction of Eq. (15) (with only constraints Eq. (5), Eq. (6) and Eq. (7)), and map it to a well-known NP-hard problem, the Multi Knapsack Problem (MKP) [1]:

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} c_i \cdot x_{i,j}, \tag{21}$$

$$s.t. \quad \sum_{i=1}^{m} x_{i,j} \leq 1, \forall j = 1, ..., n, \tag{22}$$

$$\sum_{j=1}^{n} a_i \cdot x_{i,j} \leq b_i, \forall i = 1, ..., m, \tag{23}$$

$$x_{i,j} \in \{0, 1\}, \forall i = 1, ..., m, j = 1, ..., n. \tag{24}$$

Given an instance $I = (m, n, c_i, a_i, b_i)$ of the MKP, we map it to an instance of the OIVD problem by mapping Eq. (21), (22), (23) and (24) to Eq. (15), (5), (7) and (6), and relaxing other constraints, where we get an instance of $I' = (|N| = m, |R| = n, q^{i*}_{|P^i_r|-1} = c_i, d_{r(i)} = a_i, c_{n,t'} = b_i)$. We have:

$$\max \sum_{r\in R}\sum_{i=1}^{|P_r|} q^{i*}_{|P^i_r|-1,r} \cdot \lfloor \sum_{i=1}^{|r|}\sum_{n\in N} x^{r(i)}_{n,t}/|r| \rfloor, \tag{25}$$

$$s.t. \quad \sum_{n\in N} x^{r(i)}_{n,t} \leq 1, \forall t \in T, r(i) \in F, \tag{26}$$

$$\sum_{r\in R}\sum_{i=1}^{|r|} d_{r(i)} \cdot x^{r(i)}_{n,t} \leq c_{n,t}, \forall t \in T, n \in N, \tag{27}$$

$$x^{r(i)}_{n,t} \in \{0, 1\}, \forall t \in T, r(i) \in F, n \in N. \tag{28}$$

Note that we rewrite Eq. (15) by replacing y_r with $x^{r(i)}_{n,t}$ as defined in Eq. (12) and relaxing the response latency constraint, where $y_r = 1$ equates to $\lfloor \sum_{i=1}^{|r|}\sum_{n\in N} x^{r(i)}_{n,t}/|r| \rfloor = 1$, otherwise $y_r = \lfloor \sum_{i=1}^{|r|}\sum_{n\in N} x^{r(i)}_{n,t}/|r| \rfloor = 0$. Clearly, these operations can be performed in polynomial time. Therefore, if we can solve the instance I of the MKP, we will also get a solution for the instance I' of the OIVD problem. Since MKP is NP-hard, we can conclude that the OIVD problem defined in Eq. (15) is NP-hard as well. ■

APPENDIX B

Theorem 2. *The Online Interference-aware VNF Deployment and Migration (OIVDM) problem defined in Eq. (14) is NP-hard.*

Proof: The proof process is similar to Theorem 1. Firstly, we reduce the OIVDM problem by relaxing the migration cost $M^{r(i)}_{n_1,n_2,t}$ to 0, and we have:

$$\max \sum_{t\in T}\sum_{r\in R}\sum_{i=1}^{|P_r|} q^{i*}_{|P^i_r|-1,r} \cdot y_r$$
$$s.t. \text{ (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12).} \tag{29}$$

Clearly, the relaxation can be done in polynomial time. Then, under this circumstance, since no VNF is migrated across different time slots, the total throughput of accepted requests is accumulated by summing up the throughput achieved in all time slots, where VNF deployment decisions are made in each time slot independently. Thus, the relaxation from Eq. (29) to Eq. (15) can be finished in polynomial time, and as we have proved, the relaxation from Eq. (15) to Eq. (21) can also be finished in polynomial time. Thus, we can conclude that the OIVDM problem defined in Eq. (14) is also NP-hard. ■

APPENDIX C

Theorem 3. *The worst-case performance bound of AIA is* $Z \cdot \lg(T_{max})/W$, *where* $T_{max} = \max_{r\in R} T_r$ *and* $Z = \min_{r\in R, n\in N} \prod_{i=1}^{\lfloor (|r|-1)/2 \rfloor} \alpha_{n,r(i),t} \in (0, 1]$.

Proof: Assume that Θ(AIA) represents the total throughput of accepted requests that AIA achieves, as defined in Eq. (10), while Θ(OPT) represents the optimal total throughput of accepted requests. Then, at any time slot $t \in T$, assume that in the worst case, OPT can optimally provide a non-consolidated VNF placement solution; while AIA tries to place no less than $\lfloor |P^i_r|(lg(T_r)/W) \rfloor$ VNFs in core cloud servers, which explains the first scaling step, i.e., Inequality (31). Next, since $\forall r \in R, T_{max} = \max_{r\in R} T_r \geq T_r$, we have Inequality (32). Then we rewrite $q^{i*}_{|p^i_r|-1,r}$ with the definition of $z(r, n) = q^{i*}_{j,r}/q^i_{j,r}$, and we have Inequality (33). Considering the worst case, the remaining resource of each server is not sufficient for consolidating more than two VNFs, then AIA distributes the VNFs into $\lfloor (|r| - 1)/2 \rfloor$ servers and the egress throughput of each server can not be lower than the minimized

degraded throughput (i.e., degraded by $Z = \min_{r \in R, n \in N} z(r,n)$) due to the VNF interference, and we have Inequality (34). Finally, we simplify Inequality (34) and conclude Inequality (35). As explained, $\forall t \in T$, we have:

$$\frac{\Theta(\text{AIA})}{\Theta(\text{OPT})} = \frac{\sum_{r \in R} \sum_{i=1}^{|P_r|} q^{i*}_{|p^i_r|-1,r} \cdot y_r}{\sum_{r \in R} \sum_{i=1}^{|P_r|} q^{i*}_{|p^i_r|-1,r} \cdot y^*_r} \tag{30}$$

$$\geq \frac{\sum_{r \in R} \sum_{i=1}^{|P_r|} q^{i*}_{|p^i_r|-1,r} \cdot \lg(T_r)/W}{\sum_{r \in R} \sum_{i=1}^{|P_r|} q^{i}_{|p^i_r|-1,r} \cdot 1} \tag{31}$$

$$\geq \frac{\lg(T_{max})/W \cdot \sum_{i=1}^{|P_r|} q^{i*}_{|p^i_r|-1,r}}{\sum_{i=1}^{|P_r|} q^{i}_{|p^i_r|-1,r}} \tag{32}$$

$$\geq \frac{\lg(T_{max})/W \cdot \sum_{i=1}^{|P_r|} (q^{i}_{|p^i_r|-1,r} \cdot z(r,n))}{\sum_{i=1}^{|P_r|} q^{i}_{|p^i_r|-1,r}} \tag{33}$$

$$\geq \frac{\lg(T_{max})/W \cdot \sum_{i=1}^{|P_r|} q^{i}_{|p^i_r|-1,r} \cdot Z}{\sum_{i=1}^{|P_r|} q^{i}_{|p^i_r|-1,r}} \tag{34}$$

$$\geq Z \cdot \lg(T_{max})/W \tag{35}$$

where y^*_r is the optimal solution of OPT, $z(r,n) = q^{i*}_{j,r}/q^{i}_{j,r}$. $Z = \min_{r \in R, n \in N} z(r,n) = \min_{r \in R, n \in N} \prod_{i=1}^{\lfloor (|r|-1)/2 \rfloor} \alpha_{n,r(i),t}$. ■

This worst-case performance bound indicates the largest performance gap between AIA and the optimal one-shot throughput-maximizing solution in achieving the total accepted throughput for the same set of requests as input. In practice, as evaluated in Sec. V-B of the paper, evaluation results show that AIA can improve the total throughput of accepted requests by 15.02% and 30.52% in the autonomous driving scenario and the 4K/8K HD video scenario, as compared with other state-of-the-art schemes.

REFERENCES

[1] S. Martello, "Knapsack problems: algorithms and computer implementations," *Wiley-Interscience series in discrete mathematics and optimization*, 1990.