Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices

Qixia Zhang Fangming Liu* Chaobing Zeng
Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology, China

***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 diverse performance requirements among different 5G scenarios, an adaptive VNF placement approach is needed to automatically accommodate to service-specific requirements. In this paper, we tackle the VNF placement problem by first proposing a general 5G network slice framework, which jointly contains both edge cloud and core cloud servers. Specially, based on the fact that VNF consolidation may cause severe performance degradation, we adopt a demand-supply model to quantify the VNF interference. With an aim to maximize the total throughput of accepted requests, we propose an Adaptive Interference-Aware (AIA) heuristic approach to automatically place VNFs in 5G service-customized network slices. Through simulations on two typical 5G scenarios, we demonstrate that AIA can efficiently handle traffic variation especially caused by VNF interference and improve the total throughput by 20.11% and 24.21% in autonomous driving and 4K/8K HD video network slices as compared with the state-of-the-art methods.**

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.

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 a set of independent virtual resources, independent topology, traffic of requests and its required service-specific network functions (NFs, 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) platforms [6], [7]. In this way, each network slice can perform as a logical "dedicated network" for a service-customized 5G scenario. As a result, network slicing is promised to realize the transition from one-size-fits-all to one-size-per-service manner, which can be customized for the various use cases of the 5G network. Fig. 1 depicts an example of 5G network slice infrastructure including three typical 5G use cases.

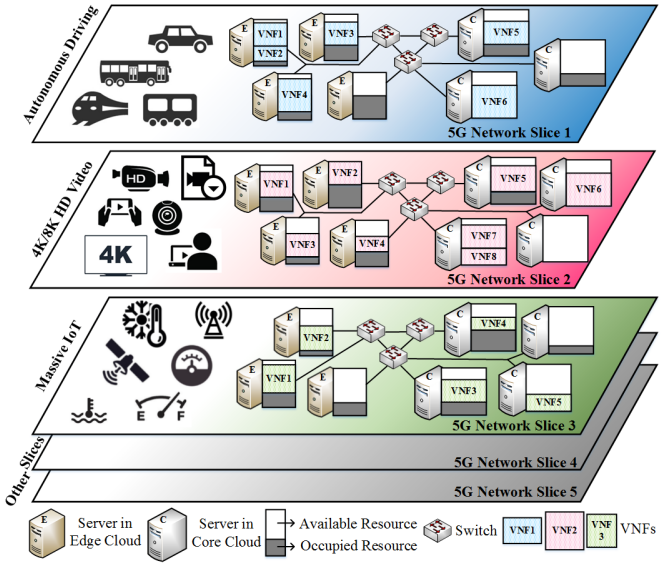


Fig. 1: 5G network slice infrastructure. In each 5G network slice, it shows a possible VNF placement solution according to the service-specific performance requirements.

Nevertheless, as illustrated in Table I [8], 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 hardwares, they are no longer restricted with fixed locations [9]. Thus, it provides an opportunity to determine how to flexibly place the service-required VNFs in each network slice so that the service-specific requirements can be satisfied and quality of service (QoS) can be further enhanced.

However, placing VNFs in service-customized 5G network slices is different from placing VNFs in traditional centralized datacenter network or Internet service provider's (ISP) network; because in some 5G use cases, we have to consider

*The corresponding author is Fangming Liu (fmliu@hust.edu.cn). This work was supported in part by the National Key Research & Development (R&D) Plan under grant 2017YFB1001703, in part by NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005, 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.

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

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 service-specific 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 the 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; while in the massive IoT network slice, smart devices intermittently generate very small amounts of data from fixed locations, their requirements on latency and bandwidth are thus not as strict as the former cases. Even though there has been some progress tackling the VNF placement problem in 5G [10]–[13], they did not take account of both edge cloud and core cloud servers in a holistic manner. Thus, their solutions can hardly meet the diverse performance requirements in various 5G scenarios.

Besides, in fact, VNFs are sometimes consolidated on the same server for the reasons of energy saving or reduction on communication latency, which is called *VNF consolidation* [14]. However, as the authors disclosed in [15], [16], VNF consolidation will cause performance degradation in terms of throughput and latency, 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 [15]. Since some 5G network slices are quite strict with the performance requirements (e.g., latency-sensitive autonomous driving and throughput-sensitive 4K/8K HD video), it is necessary to take the VNF interference into consideration. However, as far as we know, none of the existing works have captured this important characteristic when making VNF placement decisions.

Therefore, differing from previous works, we intend to propose an adaptive approach to automatically and efficiently place VNFs in 5G service-customized network slices. In particular, we firstly propose a general framework of 5G network slice, which jointly contains both edge cloud and core cloud servers with different amounts of resources (e.g., CPU and memory). Since service request is typically processed in a service function chain (SFC) [17], which is a set of VNFs that have to be orderly-executed based on the VNF forwarding graph (VNF-FG) [10], we take account of complex VNF-FGs as combinations of three basic VNF-FG topologies, i.e., linear chain, split and split-and-merged ones. To capture the VNF interference, we conduct an empirical measurement to find out the key factors that reflect the VNF interference, and then we adopt a *demand-supply model* [18] to quantify the VNF interference in terms of degraded throughput. In this way, traffic changing effects are considered in a holistic manner, including splitting and merging of flows, VNF interference and also particular network functions (e.g., compression/decompression, encryption/decryption).

With an aim to maximize the total throughput of accepted requests, we formulate the VNF placement problem in 5G network slices as an optimization problem, which is proved to be NP-hard. As plotted in Fig. 1, different 5G network slices may require different VNF placement strategies, and their performance will directly affect the QoS, resource utilization, energy consumption as well as the Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) [6]. Thus, we propose an Adaptive Interference-Aware (AIA) approach to efficiently place VNFs in service-customized 5G network slices. We derive its worst case bound and algorithm complexity. Through simulations on two typical 5G network slices (i.e., autonomous driving and 4K/8K HD video), we demonstrate that AIA can automatically fit with service-specific requirements of different 5G use cases. The main contributions are as follows:

- We propose a general 5G network slice framework for placing VNFs, which jointly contains both edge cloud and core cloud servers. In particular, we adopt a demand-supply model to quantify the VNF interference in terms of degraded throughput caused by VNF consolidation.
- With an aim to maximize the total throughput of accepted requests, we propose an Adaptive Interference-Aware approach, AIA, to automatically and efficiently place VNFs so as to accommodate to service-specific requirements in each service-customized 5G network slice.
- Through simulations on two typical 5G network slices, we demonstrate that AIA can effectively handle traffic variation especially caused by VNF interference and improve the total throughput of accepted requests by 20.11% in an autonomous driving network slice and 24.21% in a 4K/8K HD video network slice as compared with the state-of-the-art heuristic methods.

II. Related Work

In this section, we first summarize the state-of-the-art VNF placement solutions in general and then focus on the current efforts that have been paid on VNF placement in 5G. Then we explain why the existing works are not applicable to the VNF placment problem in service-customized 5G network slices.

A. VNF Placement in General

Most existing works formulate the VNF placement as an optimization problem with different objective(s) and thus a

variety of solutions are proposed, which can be classified as either exact ones or heuristic ones [6]. 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., [19]); on the contrary, the heuristic algorithms offer near-optimal solutions with short execution time, which are not limited by the network scale (e.g., [17], [20]–[22]).

In order to achieve some specific optimization objective(s) (e.g., number of servers, resource utilization, end-to-end latency and acceptance ratio of requests), the mathematical programming methods such as Integer Linear Programming (ILP) [23] and Mixed ILP (MILP) [24] are generally used. However, these solutions mainly consider placing VNFs in traditional centralized datacenter network or ISP's network, which can not capture some key features of 5G network (e.g., deploying some latency-sensitive requests in the edge cloud).

B. VNF Placement in 5G

Cao *et al.* [10] propose a two-step method for solving the VNF-FG design and VNF placement for 5G mobile networks, aiming at minimizing bandwidth consumption. Agarwal *et al.* [12] present a methodology to make joint VNF placement and CPU allocation decisions in 5G. In order to minimize the resource consumption, Alleg *et al.* [13] formulate the delay-aware VNF placement problem in 5G as a Mixed Integer Quadratically Constrained Program (MIQCP).

However, the solutions above still limit in the core cloud. Along with development of *mobile edge computing* (MEC), it is convenient to use the edge cloud to provide a wide range of services in a low-latency and energy-efficient manner [25]. Laghrissi *et al.* [26] tackle the VNF placement in dynamic and realistic edge cloud environment (i.e., edge slicing). Solozabal *et al.* [11] propose a novel energy-aware and latency-constrained VNF placement solution to edge cloud in 5G. Even though these existing works have considered using edge cloud servers to place VNFs, there still lacks a holistic solution for service-customized 5G network slices which integrally considers edge/core cloud servers and the severe VNF interference.

C. Why Are Previous Works Not Adequate for Service-Customized 5G Network Slices?

The previous works are not adequate for the VNF placement problem in service-customized 5G network slices for three main reasons. First, the general VNF placement mainly focuses on optimizing VNF placement in traditional centralized datacenter network subject to some specific objective(s), their one-size-fit-all solutions can hardly meet the diverse performance requirements in various 5G use cases. Second, some current works on VNF placement solely consider one resource type (e.g., CPU) in their model [11], [12], some consider VNF linear chains but not the general complex VNF-FGs [11], [26], [27], and most of them not integrally considers both edge and core cloud servers for deploying different requests. Last but not least, none of the aforementioned works have taken account of the ubiquitous VNF interference caused by VNF consolidation. The performance degradation can be severe and even intolerable for some QoS-sensitive 5G use cases (e.g., autonomous driving and 4K/8K HD video). Therefore, it is essential to capture this important characteristic into the model.

Differing from previous works, we propose a general framework that captures all these important features in 5G network slices, including both edge cloud and core cloud servers for placing VNFs, complex VNF-FGs, multiple types of resources (e.g., CPU and memory) and VNF interference. Based on this framework, we propose AIA, an adaptive interference-aware approach for VNF placement in 5G network slices, which can automatically and efficiently accommodate to various performance requirements of different 5G use cases.

III. A General 5G Network Slice Framework for VNF placement

In this section, we introduce the *general 5G network slice framework* for VNF placement. First, we begin with the general model of service-customized 5G network slice. Next, we conduct an empirical measurement study of consolidated VNF and then we adopt a *demand-supply model* to capture the VNF interference in terms of degraded VNF throughput. Finally, we present the mathematical formulation of VNF placement problem in 5G network slices together with objective and constraints. Key notations are listed in Table. II.

A. Model of 5G Network Slice

As stated before, a 5G network slice is a logically isolated network instance over the same underlying network, thus we represent each service-customized network slice as an undirected graph $G = (N, L)$, where N is the set of servers (nodes) and L is the set of links. To capture the important characteristics of 5G network, we use N_c to represent the set of servers in core cloud while N_e represents the servers in edge cloud. 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 $n_1, n_2 \in N, n_1 \neq n_2$.

Each server $n \in N$ has a resource capacity, representing its quantity of available resources in terms of CPU and memory, denoted by $c_n = (c_n^{cpu}, c_n^{mem})$. Note that other type of resource is relatively sufficient in servers, such as storage; while they can also be added in c_n if necessary. In a typical 5G network slice, servers in core cloud $n \in N_c$ have relatively sufficient resource capability and high response latency, while servers in edge cloud $n \in N_e$ are closer to the end-users with limited resource capability but low response latency. We denote the link latency of $l = (n_1, n_2) \in L$ by t_l^{lnk}, which includes both the propagation delay and transmission delay; while b_l represents the available bandwidth of a link $l \in L$.

We use $F = \{f_1, f_2, ..., f_{|F|}\}$ to represent the set of service-required VNFs in each service-customized 5G network slice. Each service instance of a VNF $f \in F$ has a resource demand in terms of CPU and memory, denoted by $d_f = (d_f^{cpu}, d_f^{mem})$. If a service instance of VNF $f \in F$ is deployed, it can serve

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 servers in the core cloud and N_e is the set of servers in the edge cloud
L	The set of links between the servers, $\forall l = (n_1, n_2) \in L, n_1, n_2 \in N$
F	The set of required VNFs in a 5G network slice
R	The set of requests in a 5G network slice
$A_r = (a_{r(i,j)})_{\|r\|\times\|r\|}$	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 $f_i \in F_r$ to VNF $f_j \in F_r$
$c_n = (c_n^{cpu}, c_n^{mem})$	The quantity of available resources of server $n \in N$ in terms of CPU and memory
$d_f = (d_f^{cpu}, d_f^{mem})$	The resource demand of a VNF instance $f \in F$ in terms of CPU and memory
t_f^{rsp}	The response latency of a VNF instance $f \in F$ deployed on a server for serving a request
t_l^{lnk}	The link latency on link $l \in L$
b_l	The available bandwidth resource on link $l \in L$
λ_r	The ingress throughput of request $r \in R$
T_r	The upper bound of the response latency of request $r \in R$
$x_{r(i),n}$	1 if VNF $f_i \in F_r$ of request $r \in R$ is placed on server $n \in N$, 0 otherwise
y_r	1 if request $r \in R$ is accepted, 0 otherwise

requests with an positive service rate. Thus, we define t_f^{rsp} as the processing latency of VNF $f \in F$ for serving a request.

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 an ingress throughput λ_r. As illustrated in Fig. 2, we consider three basic VNF-FG topologies: (a) *linear chain*, (b) *split* with different destinations, and (c) *split-and-merged* with a single destination. The three basic VNF-FG topologies can be further combined with one another to form more complex VNF-FGs [28]. Fig. 3 shows an example of a complex VNF-FG of request $r \in R$, $F_r = \{f_1, f_2, ..., f_7\}$. Note that:

(1) not only splitting and merging will change the throughput, some particular VNFs (e.g., compression/decompression, encryption/decryption) can also change the throughput;
(2) sometimes several replicas of the same VNF are deployed for load balancing [29], [30], we regard each replica as a separate VNF, which can equally divide the throughput.

(a) Linear Chain (b) Split (c) Split-and-merged

Fig. 2: The three basic VNF-FG topologies with an ingress point S and destination(s) D, D1 and D2.

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 flow traffic (throughput) transferring from VNF $f_i \in F_r$ to VNF $f_j \in F_r$. In addition, each request has a response latency limitation denoted by T_r.

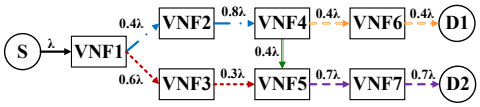
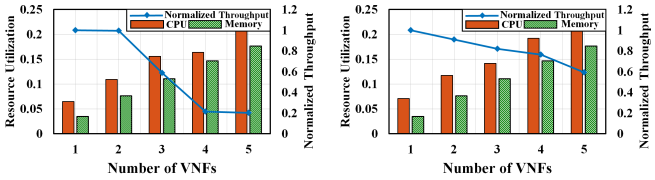


Fig. 3: An example of the VNF-FG of request $r \in R$, where λ is the ingress throughput. Note that splitting, merging and particular VNFs (i.e., VNF2 and VNF3) can all change the throughput.



(a) `Pktstat` (b) `Snort`

Fig. 4: The resource utilization of servers and the VNF's normalized throughput with different numbers of co-located VNFs.

Finally, we define a binary variable $x_{r(i),n}$ to indicate whether VNF $f_i \in F_r$ of request $r \in R$ is placed on server $n \in N$. We also define a binary variable y_r to indicate that whether request $r \in R$ is accepted or not.

B. Capturing VNF Interference

Inspired from the work in [15], 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 understand the following two questions: (1) *What is the relationship between VNF interference and the number of consolidated VNFs?* (2) *What are the key factors that reflect such VNF interference*?

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 increased on the same server.

Each server in our measurement is configured with an eight-core Intel Xeon E5-2620 v4 2.1 GHz CPU, a 64 GB memory, 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 trace 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. 4, we observe that as we scale 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 word, as more VNFs are consolidated on the same server, the resource contention is severer, which leads 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*

[1] `Pktstat`: https://linux.die.net/man/1/pktstat
[2] `Snort`: https://www.snort.org/

interference [31], 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* [18] to quantify the VNF interference in terms of degraded throughput. We devise a simple yet effective *demand-supply ratio* α_f^n to **capture the VNF interference** when VNF $f \in F$ are consolidated with other VNFs in server $n \in N$ as below,

$$\alpha_f^n = k_0 + k_1 \cdot \frac{D_n^{cpu}}{c_{n,f}^{cpu}} + k_2 \cdot \frac{D_n^{mem}}{c_{n,f}^{mem}} \tag{1}$$

where k_0, k_1 and k_2 are the coefficients in our statistical linear model, c_n^{cpu} and c_f^{mem} are the available physical CPU and memory resource "supplied" by the server $n \in N$; while $D_{n,f}^{cpu}$ and $D_{n,f}^{cpu}$ are the aggregated CPU and memory resource "demand" of all consolidated VNFs placed in server $n \in N$,

$$\forall n \in N, f \in F, D_{n,f}^{cpu} = d_f^{cpu} + \sum_{r \in R} \sum_{i=1}^{|r|} x_{r(i),n} \cdot d_{r(i)}^{cpu}. \tag{2}$$

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

This way, $D_n^{cpu}/c_{n,f}^{cpu}$ and $D_n^{mem}/c_{n,f}^{mem}$ represent the CPU and memory resource utilization, while $\alpha_f^n \in (0, 1]$ indicates the ratio of degraded throughput of VNF $f \in F$ on server $n \in N$ as compared with the throughput when f is exclusively placed on a server. Thus, the egress throughput after executing VNF $f \in F$ should be multiplied by α_f^n. The coefficients k_0, k_1 and k_2 in Eq. (1) can be acquired from extensive simulations. In our model, we initially construct a number of 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.

C. Problem Formulation

Now we formally propose the mathematical formulation of the **VNF placement problem in 5G service-customized network slices**. We begin with the constraints.

First, the total resource demand of consolidated VNFs on the same server cannot exceed its resource capacity, which can be expressed as

$$\forall n \in N : \sum_{r \in R} \sum_{i=1}^{|r|} x_{r(i),n} \cdot d_{r(i)} \leq c_n. \tag{4}$$

Next, we want to know that whether the throughput of all requests along a link $l \in L$ exceeds its link bandwidth capacity b_l. Since there might be several destinations of a request, we first figure out all the *VNF Forwarding Paths* in a VNF-FG from the ingress point to the destination(s), which we name as *VNF-FPs*. Leveraging classic graph searching algorithms such as depth-first search (DFS) [32], we can easily get a list of all the VNF-FPs of request $r \in R$ saved as $P_r = \{P_r^1, P_r^2, ...\}$, where each $P_r^i = (p_1^i, p_2^i, ...)$ stores the chain of VNFs $\forall r(p_j^i) \in F_r$ along its VNF-FP. For example, as depicted in Fig. 3, there are three VNF-FPs, P_r^1, P_r^2 and P_r^3 respectively, where $P_r^1 = (p_1^1, p_2^1, p_3^1, p_4^1) = (1, 2, 4, 6)$, $P_r^2 = (1, 2, 4, 5, 7)$ and $P_r^3 = (1, 3, 5, 7)$. Thus, for instance, $r(P_r^1) = r(1, 2, 4, 6) = (f_1, f_2, f_4, f_6)$.

For each VNF-FP $P_r^i \in P_r$, we also define a list of $Q_r = \{Q_r^1, Q_r^2, ...\}$, where each $Q_r^i = (q_1^i, q_2^i, ...)$ represents the equivalent throughput of VNF-FP $P_r^i \in P_r$ between every two VNFs, i.e., $r(p_j^i) \in F_r$ and $r(p_{j+1}^i) \in F_r$. We split and merge the throughput according to the proportion in the VNF-FG. Thus, we have:

$$\forall r \in R, i \in [1, |Q_r|], q_j^i \in Q_r^i,$$
$$q_j^i = a_{r(p_{j-1}^i, p_j^i)} \cdot \frac{\sum_{u=1}^{|r|} a_{r(p_j^i, u)}}{\sum_{v=1}^{|r|} a_{r(v, p_j^i)}} \cdot \frac{a_{r(p_j^i, p_{j+1}^i)}}{\sum_{w=1}^{|r|} a_{r(p_j^i, w)}}. \tag{5}$$

Based on Eq. (5), we can derive the equivalent throughput along each VNF-FP in Fig. 3, Q_r^1, Q_r^2 and Q_r^3 respectively, where $Q_r^1 = (q_1^1, q_2^1, q_3^1) = (0.4, 0.8, 0.4)$, $Q_r^2 = (0.4, 0.8, 0.4, 0.4)$ and $Q_r^3 = (0.6, 0.3, 0.3)$. Note that Eq. (5) does not include the VNF interference, we revise it by q_j^{i*}, where the VNF interference $\alpha_{p_j^i}^n$ is cumulated along each path.

$$q_j^{i*} = q_j^i \cdot \frac{\prod_{j=1}^{P_r^i} \prod_{n \in N} (\alpha_{r(p_j^i)}^n)^{x_{r(p_j^i),n}}}{\prod_{j=1}^{P_r^i} \prod_{n \in N} (\alpha_{r(p_j^i)}^n)^{x_{r(p_j^i),n} \cdot x_{r(p_{j+1}^i),n}}}. \tag{6}$$

Since the throughput of all requests along link $l \in L$ should not exceed its bandwidth resource capacity b_r, we represent the link resource constraint as below,

$$\forall l = (n_1, n_2) \in L, n_1, n_2 \in N,$$
$$\sum_{r \in R} \sum_{i=1}^{|r|} \sum_{j=2}^{|r|} x_{r(i),n_1} \cdot x_{r(j),n_2} \sum_{u=1}^{|Q_r|} \sum_{v=1}^{|Q_r^u|} (q_v^{u*} | p_v^u = i, p_{v+1}^u = j) \leq b_r, \tag{7}$$

where $\sum_{u=1}^{|Q_r|} \sum_{v=1}^{|Q_r^u|} (q_v^{u*} | p_v^u = i, p_{v+1}^u = j)$ represents the sum of revised throughput from VNF $r(i)$ to $r(j)$ in a request.

Next, to determine whether request $r \in R$ is accepted or not (i.e., y_r), we should check that if any VNF-FP's response latency exceeds its limitation T_r. We represent the response latency of each VNF-FP $P_r^i \in P_r$ by $\tau(P_r^i)$,

$$\tau(P_r^i) = \sum_{j=1}^{|P_r^i|} t_{r(p_j^i)}^{rsp} + \sum_{j=1}^{|P_r^i|} \sum_{n_1, n_2 \in N} t_{l=(n_1,n_2)}^{lnk} \cdot x_{r(p_j^i),n_1} \cdot x_{r(p_{j+1}^i),n_2}. \tag{8}$$

Thus, $\forall r \in R$, y_r can be represented as below,

$$y_r = \begin{cases} 1, \sum_{i=1}^{|\widehat{r}|} \sum_{n \in N} x_{r(i),n} = |\widehat{r}| \text{ and } \max_{i=1}^{|P_r|} \tau(P_r^i) \leq T_r, \\ 0, \sum_{i=1}^{|\widehat{r}|} \sum_{n \in N} x_{r(i),n} < |\widehat{r}| \text{ or } \max_{i=1}^{|P_r|} \tau(P_r^i) > T_r. \end{cases} \tag{9}$$

Our objective is to **maximize the total throughput of accepted requests**. Since we have derived the equivalent throughput q_j^{i*} along each VNF-FP $P_r^i \in P_r$, the egress

throughput of VNF-FP P_r^i can be expressed as $q^{i*}_{|P_r^i|-1}$. 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., $\sum_{i=1}^{|P_r|} q^{i*}_{|P_r^i|-1}$. Finally, the **VNF placement problem in a service-customized 5G network slice** can be formulated as

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

To achieve this objective, it is suggested to not only improve the request acceptance ratio, but also find a way to balance the trade-offs caused by VNF consolidation, i.e., the reduction on link latency and performance degradation on VNF's throughput. More importantly, as we explained previously, considering the diversity of performance requirements in different service-customized 5G network slices, an adaptive VNF placement approach is needed to automatically and efficiently accommodate to various 5G use cases.

IV. An Adaptive Interference-Aware Approach for VNF Placement

In this section, we first analysis the problem complexity of formulation Eq. (10). Then we introduce AIA, an adaptive interference-aware heuristic algorithm, which can effectively handle the VNF interference when making VNF placement decisions in different service-customized 5G network slices. Afterwards, we provide theoretical analysis of AIA in terms of optimality and algorithm complexity.

A. Problem Complexity

The VNF placement problem defined in Eq. (10) is **NP-hard**. The proof can be constructed in the same way as in [22], which reduces the problem as a variant of the NP-hard *Virtual Network Embedding* (VNE) problem. We relax the formulation of Eq. (10) with only resource constraints on servers Eq. (4) and links Eq. (7), which can be viewed as a reduction form of VNE. Since our problem is considerably harder than the VNE problem with additional constraints Eq. (1), (2), (3), (5), (6) and (8), we deduce that the VNF placement problem defined in Eq. (10) is also NP-Hard.

B. Algorithm Design

Due to the NP-hardness of Eq. (10), it is computational expensive to derive the optimal solution for a large-size network. To this end, we propose AIA, an heuristic approach which is applicable to various service-customized 5G network slices. The core concept of AIA is based on the following *four-step strategy*.

Step 1: gain the throughput of "high-yield" requests. Since there may be multiple requests arriving at a network slice simultaneously, to achieve Eq. (10), we should preferentially deploy the "high-yield" request which produces the most throughput while consuming the least resource. Thus we denote the *yield* of each request $r \in R$ by Ψ_r,

$$\Psi_r = \frac{\sum_{i=1}^{|P_r|} q^i_{|P_r^i|-1}}{\sum_{i=1}^{|r|} d_{r(i)}}, \tag{11}$$

where $\sum_{i=1}^{|P_r|} q^i_{|P_r^i|-1}$ 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 resource 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 $\tau(P_r^i), \forall\, P_r^i \in P_r$ do not exceed the response latency limitation T_r. Thus, for each request $r \in R$, it is suggested to firstly check whether its longest VNF-FP $\max \tau(P_r^i)$ 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 computation efficiency.

Step 3: adapt to various performance requirements. As we stated previously, different 5G use cases have different performance requirements in terms of latency, bandwidth, etc. While in each network slice, the requests' performance requirements are usually at the same order of magnitude. Based on these facts, we propose an adaptive method for placing the required VNFs in each service-customized 5G network slice. In particular, considering the latency requirements in different network slices, we first weight the latency limitation of request $r \in R$ by its logarithm $\lg(T_r)$. Consequently, a request with lower weight $\lg(T_r)$ means it is more latency-sensitive, which should be preferentially placed in the edge cloud and vice versa. For instance, a microsecond-level-latency request should be placed in the edge cloud with higher probability than a millisecond-level-latency one. Assume that W_t denotes the latency bound, for request $r \in R$, $\lg(T_r)/W_t$ represents the probability to use servers in the core cloud and (1 - $\lg(T_r)/W_t$) represents the probability to use servers in the edge cloud.

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. However, when doing this results in the violations of constraint Eq. (4), Eq. (7) and/or Eq. (9), then we consider consolidating two VNFs (e.g., $r(i) \in F_r$ and $r(j) \in F_r$) of request $r \in R$ if the throughput $a_{r(i,j)} \in A_r$ is the largest. This way, we reserve as much bandwidth resource as possible for deploying other VNF-FPs. Note that if consolidating two VNFs still violates the latency constraint Eq. (9), we gradually increase the number of consolidated VNFs until Eq. (9) is satisfied.

The procedure of AIA is shown in Alg. 1. We use a binary variable z_n to indicate whether server $n \in N$ has placed any VNF, and we use Ω to represent the total accepted throughput as defined in Eq. (10). We also use four sets, namely N_{e_1}, N_{e_2}, N_{c_1} and N_{c_2} 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 the current VNF, and to further distinguish whether they have

Algorithm 1 Adaptive Interference-Aware (AIA) Procedure

1: **Input**: $G = (N, L)$, R, z_n
2: **Output**: $x_{r(i),n}$, y_r, Ω
3: **Begin**: Initiate $\forall n \in N, z_n = 0$, $\Omega = 0$;
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 $P_r^i \in P_r$ by its $|P_r^i|$ in descending order;
7: **while** $P_r \neq \emptyset$ **do**
8: Get P_r^i with the maximal $|P_r^i|$;
9: **for** $j = 1$ to $|P_r^i|$ **do**
10: **if** $\sum_{n \in N} x_{r(p_j^i),n} = 1$ **then**
11: $j = j + 1$, continue;
12: **else**
13: $f_j = r(p_j^i)$;
14: **if** $N_{c_1} \cup N_{e_2} \cup N_{c_2} \cup N_{c_1} = \emptyset$ **then**
15: $y_r = 0$, $R = R - \{r\}$, deploy the next $r \in R$;
16: **else if** $j \leq \lceil (|P_r^i|(1 - \lg(T_r)/W_t)) \rceil$ **then**
17: Preferentially place f_j at $n \in N_{e_1}$ with larger
18: c_n, then orderly place in N_{c_1}, N_{e_2} and N_{c_2};
19: **else**
20: Preferentially place f_j at $n \in N_{c_1}$ with higher
21: c_n, then orderly place in N_{e_1}, N_{c_2} and N_{e_2};
22: **end if**
23: **if** Eq. (7) = 1 **then** $x_{f_j,n} = 1$, $z_n = 1$, j++;
24: **else** Place f_j at other server n orderly in N_{e_1}, N_{c_1},
25: N_{e_2} and N_{c_2} with larger c_n until Eq. (7) =1;
26: $x_{f_j,n} = 1$, $z_n = 1$, j++;
27: **end if**
28: **if** $j = |P_r^i|$ **then**
29: **while** $\tau(P_r^i) > T_r$ **do**
30: Consolidate VNFs in P_r^i with the largest
31: throughput q_j^i, then update each q_j^{i*} ;
32: **if** $\tau(P_r^i) \leq T_r$ **then** $\Omega += q_{|P_r^i|-1}^{i*}$,
33: $P_r -= \{P_r^i\}$, deploy VNF-FP P_r^{i+1};
34: **end if**
35: **end while**
36: $y_r = 0$, $R = R - \{r\}$, deploy the next $r \in R$;
37: **end if**
38: **end if**
39: **end for**
40: **end while**
41: **end while**

placed any VNF (i.e., N_{e_1}, N_{c_1}) or not (i.e., N_{e_2}, N_{c_2}) for the further consideration on the VNF interference.

C. Optimality Analysis

Now we give a brief optimality analysis of AIA.

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

Proof: Assume that Θ(AIA) represents the total throughput of accepted requests of AIA as defined in Eq. (10), while Θ(OPT) represents the optimal total throughput of accepted requests. Then, assume that in the worst case, OPT can optimally provide non-consolidated VNF placement solution; while AIA tries to place $\lfloor |P_r^i|(lg(T_r)/W_t) \rfloor$ VNFs in the core cloud, which may lead to violations on constraint Eq. (9) and get $y_r = 0$. In this case, the remaining resource in each server is not sufficient for consolidating more than two VNFs. Thus, AIA distributes the VNFs into $\lfloor (|r| - 1)/2 \rfloor$ servers and the egress throughput of each server is degraded by $\alpha_{f(i)}^n$ as a result of VNF consolidation. Thus, we have:

$$\begin{aligned}
\frac{\Theta(\text{AIA})}{\Theta(\text{OPT})} &\geq \frac{\sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|p_r^i|-1}^{i*} \cdot y_r}{\sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|p_r^i|-1}^{i} \cdot y_r^*} \\
&\geq \frac{\sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|p_r^i|-1}^{i*} \cdot \lg(T_r)/W_t}{\sum_{r \in R} \sum_{i=1}^{|P_r|} q_{|p_r^i|-1}^{i} \cdot 1} \\
&\geq \frac{\lg(T)/W_t \cdot \sum_{i=1}^{|P_r|} (q_{|p_r^i|-1}^{i} \cdot z(r,n))}{\sum_{i=1}^{|P_r|} q_{|p_r^i|-1}^{i}} \\
&\geq Z \cdot \lg(T)/W_t
\end{aligned}$$

where y_r^* is the optimal solution of OPT, $z(r,n) = q_j^{i*}/q_j^i$. ■

D. Complexity Analysis

We now analyze the complexity of our proposed algorithm, AIA. First, 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 $\mathbf{M} = \max_{i=1} |P_r^i|$ iterations and finally the last **while**-loop (line 29) terminates in $\mathbf{N} = \max_{i=1}(|P_r^i| - 1)$ iterations. Since $\log_2 \mathbf{R} < \mathbf{LMN}$, $\log_2 \mathbf{L} < \mathbf{MN}$, the time complexity of AIA is $O(\mathbf{R} \log_2 \mathbf{R} + \mathbf{R}(\mathbf{L} \log_2 \mathbf{L} + \mathbf{LMN})) = O(\mathbf{RLMN})$.

V. Performance Evaluation

To demonstrate the effectiveness of our proposed algorithm AIA, we simulate two typical 5G use cases, i.e., **CASE I: an autonomous driving network slice** and **CASE II: a 4K/8K HD video network slice**.

A. Simulation Setup

Network slice topology: we use the `GT-ITM`[3] tool to generate the 5G network topology. In each 5G network slice, we divide the edge/core cloud servers by their distances from the end-users as shown in [33], and we scale the number of core cloud servers from 5 to 300 and double the number of edge cloud servers scaling from 10 to 600. According to [22], 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] respectively. The CPU and memory resource required by each VNF are randomly distributed in [1, 20] units and [1, 4] units.

Parameter of 5G use cases: due to the diverse performance requirements among different 5G use cases, the service-customized 5G network slices are instantiated with different configurations. As shown in Table. III, we set the values of parameters based on [8], [26], [34]. The VNF-FGs are generated in the similar way as stated in [21], including the three basic VNF-FG topologies and their combinations as we elaborate in Sec. III-A. In both 5G use cases, the number of requests that we simulate ranges from 5 to 1000.

Algorithm compared: we compare AIA with the following two state-of-the-art heuristic algorithms, which are introduced in detail in [35].

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

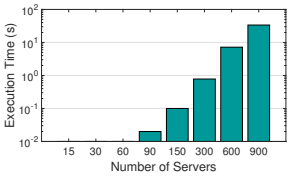


Fig. 5: The execution time with different number of servers.

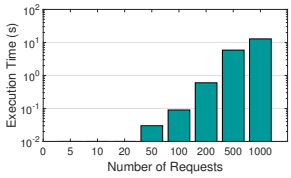


Fig. 6: The execution time with different number of requests.

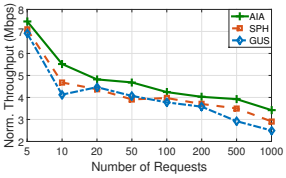


Fig. 7: The normalized total throughput of accepted request in the autonomous driving case.

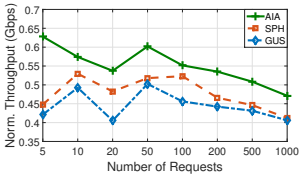


Fig. 8: The normalized total throughput of accepted request in the 4K/8K HD video case.

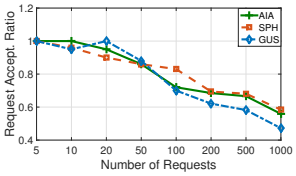


Fig. 9: The request acceptance ratio in the autonomous driving network slice.

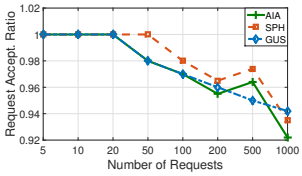


Fig. 10: The request acceptance ratio in the 4K/8K HD video network slice.

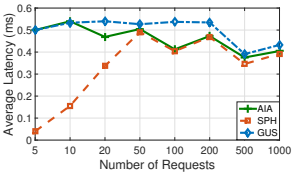


Fig. 11: The average response latency in the autonomous driving network slice.

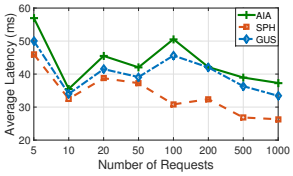


Fig. 12: The average response latency in the 4K/8K HD video network slice.

TABLE III: Key Notations

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	NAT, FW, TM, VOC, IDCS, CNF, DNF
Number of P_r of each VNF-FG	[1, 3]	[1, 6]
t_f^{rsp}	0.1 ms	1 ms
t_l^{lnk}	0.01∼0.5 ms	1∼5 ms
b_l	100 Mbps, 1 Gbps	1 Gbps, 10 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 requried Network Function, VOC: Video Optimization Controller, CNF: Compression Network Function, DNF: Decompression Network Function

- **Shortest Path Heuristic (SPH)**: it deploys each VNF-FP of a request along the shortest path between the ingress point and the egress point(s). SPH preferentially places and consolidates VNFs in edge servers so as to reduce the response latency along the links.
- **Greedy on Used Server (GUS)**: it aims at minimizing the number of used servers so as to reduce the OPEX and energy consumption, which preferentially places VNFs in the used servers to avoid occupying extra unused servers.

B. Performance Evaluation Results

Execution time. Based on the previous analyses, we know that AIA's time complexity is positively correlated with the number of servers and requests. Thus, we scale the number of servers and requests to demonstrate the efficiency of AIA. Fig. 5 depicts the execution time of AIA with the number of total servers scaling from 15 to 900. 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., 33.69 seconds for a network with 900 servers). Fig. 6 shows that AIA can converge in 1 second for handling less than 200 requests. Even for dealing with 1000 requests, AIA can find a solution in 12.84 seconds. Thus, we can conclude that 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 (i.e., Ω, the objective), as we defined in Eq. (10). For highlighting the effectiveness, we depict the normalized throughput (i.e., the total throughput divided by the number of input requests) of the autonomous driving network slice (CASE I) and the 4K/8K HD video network slice (CASE II) respectively in Fig. 7 and Fig. 8. As the number of requests scales from 5 to 1,000, the normalized total throughput increases accordingly; while AIA achieves averagely 12.30% and 20.11% more total throughput of accepted requests than SPH and GUS in CASE I, and correspondingly 15.58% and 24.21% in CASE II. In fact, AIA improves the total accepted throughput by 17.60% and 37.05% when dealing with 1,000 latency-sensitive requests in the autonomous driving case as compared with SPH and GUS. This benefits from the *four-step strategy* as we detailed in Sec. IV. Especially, AIA reduces the VNF interference in terms of throughput by decreasing the probability of VNF consolidation. As a result, AIA can effectively handle traffic variation especially caused by VNF interference and always achieve the maximal total throughput of accepted requests in both typical 5G use cases.

Request acceptance ratio. Next, we compare the ratio of accepted requests among all requests in both cases, which are plotted in Fig. 9 and Fig. 10 respectively. In the autonomous driving case, the request acceptance ratio decreases notably as the number of requests increases, i.e., from 100% to 55.9%. 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 the 4K/8K HD video case, the request acceptance ratio decreases slightly, i.e., from 100% to 92.20%. This is because the response latency requirement is not as strict as CASE I. The average request

acceptance ratios of AIA, SPH, GUS are respectively 97.39%, 98.18%, 97.53% in CASE I and 80.50%, 81.36%, 77.56% in CASE II. Note that even though SPH's request acceptance ratio is a little bit higher than AIA, i.e., around 2%, its total throughput is averagely 14% less than AIA.

Average response latency. Finally, we also compare the average response latency of accepted requests. As plotted in Fig. 11 and Fig. 12, the average response latency fluctuates around 0.4∼0.5 ms in the autonomous driving case, and fluctuates around 30∼50 ms in the 4K/8K HD video case. Note that when the number of requests is less than 50, SPH shows a good advantage in reducing the response latency in CASE I. This is because SPH consolidates as many VNFs as possible in the edge cloud so that some requests can be deployed without using servers in the core cloud. However, this can inevitably cause throughput degradation as shown in Fig. 7 and Fig. 8. Even though SPH outperforms AIA and GUS a little bit in response latency, AIA can also meet the latency requirements in both 5G use cases.

VI. Conclusion

In this paper, we present the study on VNF placement in 5G service-customized network slices. We propose a general 5G network slice framework, which jointly contains both edge cloud and core cloud servers. Through empirical measurement study of consolidated VNFs, we use a demand-supply model to quantify the ubiquitous VNF interference. We prove the NP-hardness of the problem formulation. Then we propose AIA, an efficient adaptive interference-aware heuristic approach to automatically place VNFs in 5G network slices. We also derive its worst-case performance bound and algorithm complexity. Through simulations on two typical 5G scenarios, we demonstrate that AIA can effectively handle traffic variation especially caused by VNF interference and improve the total throughput of accepted request by 20.11% in the autonomous driving network slice and 24.21% in 4K/8K HD video network slice as compared with other heuristics.

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] Recently and F. Example, "An optimal routing algorithm in service customized 5G networks," *Mobile Information Systems, 2016-1-4*, vol. 2016, pp. 1–7, 2016.

[6] 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.

[7] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM*, 2018, pp. 486–494.

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

[9] W. Ma, O. Sandoval *et al.*, "Traffic aware placement of interdependent NFV middleboxes," in *IEEE INFOCOM*, 2017, pp. 1–9.

[10] 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.

[11] R. Solozabal, B. Blanco *et al.*, "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.

[12] S. Agarwal, F. Malandrino *et al.*, "Joint VNF placement and CPU allocation in 5G," in *IEEE INFOCOM*, 2018, pp. 1943–1951.

[13] A. Alleg, T. Ahmed *et al.*, "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.

[14] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "Netbricks: Taking the v out of NFV." in *12th USENIX Symposium OSDI*, 2016, pp. 203–216.

[15] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the performance interference of co-located virtual network functions," in *IEEE INFOCOM*, 2018, pp. 765–773.

[16] A. Tootoonchian, A. Panda *et al.*, "Resq: Enabling slos in network function virtualization," in *15th USENIX Symposium on NSDI*, 2018.

[17] Q. Zhang, Y. Xiao, F. Liu, J. C. 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.

[18] 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*, vol. 63, no. 12, pp. 3012–3025, 2014.

[19] M. Shifrin, E. Biton, and O. Gurewitz, "Optimal control of VNF deployment and scheduling," in *IEEE ICSEE*, 2016, pp. 1–5.

[20] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM*, 2017, pp. 1–9.

[21] 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.

[22] 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.

[23] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Global Communications Conference*, 2017.

[24] T. Lin, Z. Zhou *et al.*, "Demand-aware network function placement," *Journal of Lightwave Technology*, vol. 34, no. 11, pp. 2590–2600, 2016.

[25] Q. Xu, D. Gao *et al.*, "Low latency security function chain embedding across multiple domains," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.

[26] A. Laghrissi, T. Taleb *et al.*, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *IEEE GLOBECOM*, 2018, pp. 1–6.

[27] Y. Nam, S. Song, and J. M. Chung, "Clustered NFV service chaining optimization in mobile edge clouds," *IEEE Communications Letters*, vol. PP, no. 99, pp. 1–1, 2016.

[28] M. C. Luizelli, L. R. Bays *et al.*, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *IFIP/IEEE International Symposium on Integrated Network Management*, 2015, pp. 98–106.

[29] T. Wang, H. Xu, and F. Liu, "Multi-resource load balancing for virtual network functions," in *IEEE ICDCS*, 2017, pp. 1322–1332.

[30] X. Fei, F. Liu, H. Xu, and H. Jin, "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure," in *IEEE/ACM International Symposium on Quality of Service*, 2017, pp. 1–10.

[31] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2470–2483, 2016.

[32] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[33] R. Cziva, C. Anagnostopoulos *et al.*, "Dynamic, latency-optimal vNF placement at the network edge," in *IEEE INFOCOM*, 2018.

[34] S. B. Mer, "Advanced autonomous vehicle with 5G technology," *Int. J. Eng. Develop. Res.*, vol. 3, no. 2, pp. 2321–9939, 2015.

[35] T. W. Kuo, B. H. Liou *et al.*, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM*, 2016, pp. 1–9.