Joint Virtual Network Function Placement and Flow Routing in Edge-Cloud Continuum

Yingling Mao, *Graduate Student Member, IEEE*, Xiaojun Shang, Yu Liu, *Graduate Student Member, IEEE*, and Yuanyuan Yang, *Life Fellow, IEEE*

***Abstract*—Network Function Virtualization (NFV) is becoming one of the most popular paradigms for providing cost-efficient, flexible, and easily-managed network services by migrating network functions from dedicated hardware to commercial general-purpose servers. Despite the benefits of NFV, it remains a challenge to deploy Service Function Chains (SFCs), placing virtual network functions (VNFs) and routing the corresponding flow between VNFs, in the edge-cloud continuum with the objective of jointly optimizing resource and latency. In this paper, we formulate the SFC Deployment Problem (SFCD). To address this NP-hard problem, we first introduce a constant approximation algorithm for a simplified SFCD limited at the edge, followed by a promotional algorithm for SFCD in the edge-cloud continuum, which also maintains a provable constant approximation ratio. Furthermore, we provide an online algorithm for deploying sequentially-arriving SFCs in the edge-cloud continuum and prove the online algorithm achieves a constant competitive ratio. Extensive simulations demonstrate that on average, the total costs of our offline and online algorithms are around 1.79 and 1.80 times the optimal results, respectively, and significantly smaller than the theoretical bounds. In addition, our proposed algorithms consistently outperform the popular benchmarks, showing the superiority of our algorithms.**

***Index Terms*—Network function virtualization, service function chain deployment, edge computing, cloud computing, joint resource and latency optimization.**

I. Introduction

NETWORK Function Virtualization (NFV) is a promising technique that enables the migration of network functions such as Proxies, Firewalls, Load Balancers, etc, from dedicated hardware to commercial servers. It brings flexibility, scalability, and cost-efficiency to network services. These virtual network functions (VNFs) are typically chained together in a specific order to form service function chains (SFCs) [1]. With the growing development of low-latency edge computing, there is increasing motivation to deploy SFCs at the edge [2], [3], [4], [5], [6], [7], [8], [9]. Deploying SFCs at the network edges offers numerous benefits, such as reducing communication delays, avoiding network congestion, and enhancing data safety and privacy [10].

Although NFV and edge computing offers many benefits, deploying SFCs at the edge can be challenging. On the one hand, efficient resource management is crucial due to the limited and expensive nature of edge resources. Specifically, the task is to place VNFs on as fewer edge commercial servers as possible under server capacity constraints. On the other hand, we must carefully schedule the data flow between adjacent VNFs during SFC deployment. Poor scheduling schemes can result in redundant flow paths, leading to network congestion and high latency, while well-designed schemes can significantly reduce communication latency. Therefore, our model aims to jointly optimize resources and latency, efficiently managing edge resources and reducing communication latency simultaneously.

Several existing studies exclusively deploy SFCs on edge servers [2], [3]. However, there are extreme scenarios where deploying all SFCs on edge with the limited edge resources is infeasible. For example, during peak hours [11], the majority of users are requesting services, and the edge resources are incapable of hosting all requesting SFCs. Under such scenarios, we can offload some SFCs to the cloud which can provide sufficient computing resources [12]. Some other existing studies adopt a trivial approach of placing all remaining VNFs on the remote cloud once all edge servers are fully utilized [4], [5]. However, this approach can result in increased communication latency because SFCs consist of diverse VNFs and offloading different SFCs to the cloud can significantly impact the overall communication latency. Besides, as for some special VNFs, i.e. Low-Density Parity Check (LDPC) or Turbo decoding, cloud servers can provide more powerful computing resources like GPU or FPGA to cut back the processing latency. Thus, there is a tradeoff between communication and processing latency when employing cloud resources. Therefore, it is necessary to overall consider the SFC deployment problem in the edge-cloud continuum.

In all, we formulate and study the SFC Deployment (SFCD) problem, i.e., jointly placing VNFs in the edge-cloud continuum and routing the data flow between VNFs, with the goal of

Manuscript received 24 April 2023; revised 12 December 2023; accepted 19 December 2023. Date of publication 28 December 2023; date of current version 12 February 2024. This work was supported in part by the U.S. National Science Foundation under Grants CCF-1526162, CCF-1730291, and CCF-1717731. Recommended for acceptance by X. Fu. *(Corresponding author: Yuanyuan Yang.)*

Yingling Mao, Yu Liu, and Yuanyuan Yang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794 USA (e-mail: yingling.mao@stonybrook.edu; yu.liu.3@stonybrook.edu; yuanyuan.yang@stonybrook.edu).

Xiaojun Shang is with the Department of Computer Science and Engineering at the University of Texas, Arlington, TX 76019 USA (e-mail: xiaojun.shang@uta.edu).

Digital Object Identifier 10.1109/TC.2023.3347671

optimizing resource cost and network latency jointly. The problem is challenging for the following reasons. First, the problem involves balancing the trade-off between resource cost, and network latency, i.e., communication and processing latency, which can be contradictory objectives, referring to work [13]. Second, incorporating edge and cloud makes the problem more complicated. Additionally, the problem is NP-hard due to the integer-variable constraint arising from the indivisible VNF and limited edge server capacity. Furthermore, the complex network topology poses difficulties in virtual network embedding, especially in solving the routing problem of data flow between VNFs. Last but not least, we pursue to design a provable approximation algorithm for SFCD with an excellent theoretical bound, which is also a big challenge.

In this paper, we design both offline and online algorithms for the SFCD problem. We first simplify the SFCD problem to an edge-only scenario where edge resources are sufficient for all SFCs and thus only edge resources are considered. We propose a constant approximation algorithm called Chained Next Fit (CNF) for the edge-only cases. The CNF algorithm is based on the Next Fit (NF) strategy [14] and utilizes a double spanning tree (DST) algorithm to handle the network topology and the corresponding flow routing problem. The NF strategy can guarantee efficient resource utilization while avoiding redundant data traffic, and the DST algorithm helps to reduce latency and prevent network congestion. Subsequently, we present the Edge-First Chained Next Fit (ECNF) algorithm, a enhanced version of the CNF algorithm that addresses the general SFCD problem in the edge-cloud continuum overall considering both edge and cloud resources. Additionally, we demonstrate that ECNF can be bounded by a provable constant approximation ratio. Furthermore, we propose an online algorithm named Online Chained Next Fit (OCNF) to address the online version of SFCD, where SFCs sequentially arrive in an online manner. OCNF is also proved to be a constant approximation algorithm.

Our main contributions are summarized as follows.

- We formulate the SFCD problem, which jointly considers resource cost and network latency in the edge-cloud continuum.
- We propose a constant approximation algorithm called CNF for the edge-only cases of the SFCD problem.
- Based on CNF, we propose an enhanced algorithm named ECNF to address the general SFCD problem overall considering edge and cloud. Also, we demonstrate that the ECNF algorithm has a constant approximation ratio.
- To address the online SFCD problem, where SFCs arrive sequentially in an online manner, we propose an online algorithm named OCNF and prove that it has a provable performance guarantee.
- Finally, we conduct extensive simulations using real-world network topologies to evaluate the proposed algorithms. Simulation results are consistent with the theoretical performance bounds and demonstrate that the proposed algorithms outperform the baselines.

The remainder of this paper is organized as follows. Section II briefly reviews the related work. In Section III, we give the formulation of the SFCD problem. Section IV demonstrates our CNF algorithm designed for the simplified SFCD limited at the edge and proves its constant theoretical bound. In Section V, CNF is generalized to ECNF for completely solving SFCD in the edge-cloud continuum. Section VI handles the online version of SFCD and provides an online algorithm, called OCNF. Additionally, Section VII is the performance evaluation of ECNF and OCNF. Finally, we conclude the paper in Section VIII.

II. Related Work

With the emergence of NFV, researchers have devoted much effort to SFC deployment problems like [2], [3], [4], [5], [6], [7], [8], [9], [13], [15], [16], [17], [18], [19], [20], [21], [22]. In this research area, there is a trend of deploying SFCs on commercial servers at the edge because of the low latency of edge computing.

Most of the existing related work is devoted to designing heuristic algorithms for deploying SFCs on edge. For example, Cziva et al. [2] presented a way to dynamically re-schedule the optimal placement of vNFs at the network edge, based on temporal network-wide latency fluctuations using optimal stopping theory. Pei et al. [3] proposed the novel SFC embedding approach (SFC-MAP) and VNF dynamic release algorithm (VNF-DRA) to efficiently embed SFC requests in geo-distributed cloud systems and optimize the number of placed VNF instances. Son et al. [4] proposed a dynamic resource provisioning algorithm. It automatically allocates resources in both the edge and the cloud for VNFs, adapting to dynamically changing network volumes. Martín-Peréz et al. [5] presented a novel methodology and resource allocation scheme, named OKpi, which enables high-quality selection of radio points of access as well as VNF placement and data routing with polynomial computational complexity. He et al. [23] leveraged the Markov Decision Process to model the dynamic network states and devised a customized Deep Reinforcement Learning (DRL) algorithm for the VNF placement problem. These works are limited to the design of heuristic algorithms and do not have a provable performance guarantee.

As far as we are concerned, there are only five related works giving the performance bounds. Sang et al. [18] designed two simple greedy algorithms and proved that they achieve an asymptotical approximation ratio of $(1-0(1))lnm+2$, where m is the number of flows. Jin et al. [6] designed a two-stage VNF deployment scheme, including a constrained depth-first search algorithm (CDFSA) and a path-based greedy algorithm (PGA), to deploy VNF chains at network edges with latency guarantees and resource efficiency. It gives a theoretically-proved worst-case performance bound by an implicit constant factor. In [8], Mao et al. produced the judge and repeated largest fit decreasing algorithm (JR-LFD) with an asymptotic approximation ratio of $\frac{3}{2}$ to deploy VNFs at network edges. Ren et al. [7] discussed a fundamental problem of NFV-enabled multicasting in a mobile edge cloud, devised an approximation algorithm with a provable approximation ratio for a single multicast request admission if its delay requirement is negligible. In [9], an efficient randomized rounding approximation algorithm was proposed to solve the delay-aware virtual network function placement and routing in the edge-and-cloud network.

These five theoretically provable works all have some drawbacks to their model and their performance guarantees only work for the specific simplified model. For example, work [6], [8], [18] all limit to the edge-only cases. Since the resource of edge computing is limited, it is likely to occur the case that there is no way to deploy all SFCs on the edge servers under the server capacity limitation. Thus, it is necessary to introduce the cloud node into the network model, considering the SFC deployment schemes in the edge-cloud continuum. Work [7], [8], [18] only minimizes the edge server resource consumption, ignoring the network latency, while [9] only optimizes the network latency, ignoring the resource cost. In this paper, we consider a general model, which targets joint resource and latency optimization in the edge-cloud continuum. Under this general model, the performance guarantee of existing provable works may break. Thus, it is the first work to pursue a provable algorithm for the joint VNF placement and flow routing in the edge-cloud continuum.

III. Problem Formulation

A. Background

Network functions play a pivotal role in shaping the dynamics of a network infrastructure. A diverse array of network functions, including Proxies, Firewalls, Load Balancers, and others, form the backbone of various network services. Traditionally, these functions are implemented on proprietary hardware, incurring high costs and posing challenges in terms of management and upgrades. The emergence of virtualization technologies introduced the concept of "Network Functions Virtualisation" (NFV) in 2012 [24], presenting a novel solution to address these challenges. NFV embodies a software-defined network (SDN) architecture, wherein virtual network functions (VNFs) are instantiated on virtual machines (VMs) hosted on commercial servers.

A network service normally consists of multiple VNFs, which exhibit dependencies on one another. For example, in the case of a WAN optimizer and an Intrusion Detection System (IDS), the IDS typically conducts packet inspection prior to the WAN optimizer encrypting the contents. In sum, the realization of a network service necessitates a sequence of VNFs, orchestrated in a specific order. We call this sequence of VNFs a service function chain (SFC) [1].

SFC deployment problem involves strategically placing these VNFs within the edge-cloud environments and efficiently routing the data flows between inter-dependent VNFs within an SFC to ensure efficient resource utilization and low network latency. Below we will develop a model designed to address the dual optimization goals of resources and latency in the SFC deployment problem.

B. System Model

Table I summarizes the notations used in this paper.

We consider a edge network represented as a connected graph $G = (V, E)$, where every commercial server i is a vertex in the graph denoted as V_i. If server V_p and V_q are directly connected in the edge network, then the link $(p, q) \in E$, otherwise, $(p, q) \notin E$. We assume that there are M servers located at the edge and that the processing capacity of server i is C_i. As shown in Fig. 1, we assume there exists a central server in the edge network, which is "directly"[1] connected to the remote cloud via ISP router. We note such a special server as V_1.

We assume there are m SFCs requested by users, where each SFC comprises an ordered set of VNFs, and the data throughput of SFC i is b_i. SFC i contains n_i VNFs, noted as $F_{i,1}, F_{i,2}, \cdots, F_{i,n_i}$ in chaining order, so the total number of VNFs is $N = \sum_{i=1}^{m} n_i$. Suppose VNF $F_{i,j}$ requires $f_{i,j}$ computing resource, i.e., the size of $F_{i,j}$ is $f_{i,j}$. However, the processing time of the same VNF may differ on edge and cloud. Because edge nodes may only contain basic CPU resources, while today's cloud nodes may contain powerful GPU or FPGA, which can accelerate some specific VNFs, e.g., Low-Density Parity Check (LDPC) or Turbo decoding can rely on GPU to

TABLE I
Notations

m	number of Service Function Chains (SFCs)
M	number of physical servers in the edge network
n_i	number of VNFs in SFC i
N	number of all VNFs
$F_{i,j}$	VNF j in SFC i
$f_{i,j}$	the size of VNF $F_{i,j}$
l_i	the one-hop communication latency of SFC i
L_i	outside-edge (E&C) communication latency of SFC i
b_i	data flow of SFC i
V_k	edge server k, if $k > 0$; the cloud, if $k = 0$
C_k	capacity of edge server k
$B_{p,q}$	bandwidth limit of network connection (p, q)
$w_{i,j}^{p,q}$	if data between $F_{i,j}$ and $F_{i,j+1}$ pass link (p, q) or not
$x_{i,j}^{k}$	if VNF $F_{i,j}$ is placed on V_k or not
y_k	if server k is occupied or not
$z_{i,j}$	num. of hops at edge when data flows from $F_{i,j}$ to $F_{i,j+1}$
$Z_{i,j}$	if there is E&C com. latency between $F_{i,j}$ and $F_{i,j+1}$ or not
M'	number of used servers in the results of CNF
M^*	number of used servers in OPT
V_j^*	used server j in OPT
$C(\cdot)$	(total) capacity of a server
$c(\cdot)$	occupied capacity of a server by VNFs.
z_i	number of hops in SFC i by CNF
z_i^*	number of hops in SFC i in OPT
$t_{i,j}$	the processing time of VNF $F_{i,j}$ on edge nodes
$T_{i,j}$	the processing time of VNF $F_{i,j}$ on cloud

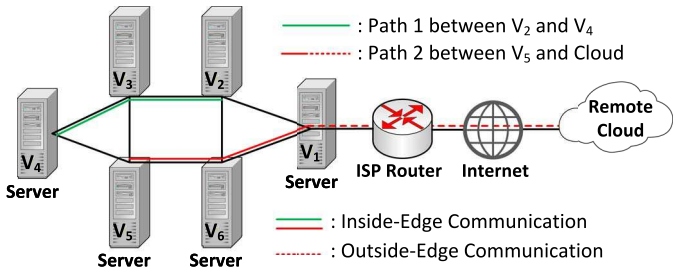


Fig. 1. System topology.

[1]The "directly" here means the central server is connected directly to the ISP (Internet Service Provider) router, which helps the edge servers communicate to the cloud. If other servers want to communicate to the cloud, the data flow must pass through this server.

achieve lower processing time. Thus, we assume the processing time of VNF $f_{i,j}$ is $t_{i,j}$ on edge nodes and $T_{i,j}$ on the cloud, where $T_{i,j} \leq t_{i,j}$.

In each SFC i, there is a data flow between $F_{i,j}$ and $F_{i,j+1}$ for any $1 \leq j \leq n_i - 1$. If VNF $F_{i,j}$ and $F_{i,j+1}$ are placed on different edge commercial servers, transmitting data from $F_{i,j}$ to $F_{i,j+1}$ incurs communication latency. Since G is usually not a complete graph, data transmission from $F_{i,j}$ on one server to $F_{i,j+1}$ on another server may need to traverse several server-nodes in G. The communication latency here is determined jointly by the one-hop communication latency of SFC i, noted as l_i, and the number of hops or network connections that the data flow passes through, denoted as $z_{i,j}$. The number of hops $z_{i,j}$ depends on the solutions of the multi-hop routing problem on G under the bandwidth limit of each link (p, q), noted as $B_{p,q}$. Note that the network topology information is also contained in $\{B_{p,q}\}$, i.e., if $(p, q) \notin E$, $B_{p,q} = 0$.

We discussed the communication latency at the edge above. However, if some VNFs are offloaded to the cloud, communication latency between the edge and cloud arises, referred to as E&C communication latency. We consider a scenario where VNF $F_{i,j}$ and $F_{i,j+1}$ are placed on the edge and cloud, respectively, resulting in E&C communication latency. Since most servers, except the central server, can not "directly" communicate with the cloud, such communication latency typically has two parts like the red line in Fig. 1, the inside-edge part and the outside-edge part. The inside-edge E&C communication latency, for example the red solid line in Fig. 1, is exactly the communication latency between the targeted server V_5 and the central server at edge V_1, which can be computed by $l_i \cdot z_{i,j}$ as above mentioned. The outside-edge E&C communication latency, for example the red dotted line in Fig. 1, depends on various factors, including the SFC data flow, multiple hops outside the edge network, and bandwidths between the edge and cloud, and many other factors like network environment along the path outside the edge, making it a complex issue [12]. For simplicity, here we just assume the outside-edge E&C communication latency, short for outside-edge communication latency, is L_i with $L_i \geq l_i$. The cloud is denoted as V_0. Since only the central server V_1 is "directly" connected to the cloud V_0, here we give the estimated bandwidth limit of the connection between the central server V_1 and the cloud V_0, noted as $B_{1,0} = B_{0,1}$. Similarly as the case at the edge network, here we also hide the network topology information between edge and cloud in $\{B_{0,p}, B_{p,0}\}$. That is, for any non-central server $V_p \in V (p > 1)$ that is not "directly" connected to the cloud V_0, $B_{p,0} = B_{0,p} = 0$.

C. Problem Formulation

The SFCD problem is to place the m SFCs onto the M edge commercial servers or the cloud without exceeding the constraints of edge server capacities and the bandwidth limits of network connections. The goal of the problem is to minimize the total resource consumption and produced network latency in the SFC deployment scheme. In particular, the total resource consumption contains the total capacities of occupied edge servers and the total sizes of VNFs offloaded to the cloud. On the other hand, the total network latency includes the total processing latency, the communication latency between VNFs placed at the edge, and that between the edge and cloud.

To formulate SFCD, we define three *Boolean* variables $x^k_{i,j}$, y_k, $w^{p,q}_{i,j}$, where $x^k_{i,j}$ and $w^{p,q}_{i,j}$ are the decision variables in our model while y_k is used for clear expression. Note that $x^k_{i,j} = 1$ if and only if VNF $F_{i,j}$ is placed on server V_k; $y_k = 1$ if and only if server V_k is occupied; $w^{p,q}_{i,j} = 1$ if and only if data flow between VNF $F_{i,j}$ and $F_{i,j+1}$ pass through network connection between V_p and V_q, where $(V_p, V_q) \in E$.

First, the capacity constraint of each edge server asks

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} x^k_{i,j} \cdot f_{i,j} \leq y_k \cdot C_k, \quad \forall 1 \leq k \leq M. \tag{1}$$

And the limitation of bandwidth requires

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i - 1} \left(w^{p,q}_{i,j} + w^{q,p}_{i,j} \right) \cdot b_i \leq B_{p,q}, \quad \forall\, 0 \leq p, q \leq M. \tag{2}$$

It is worth noticing that $B_{p,q} = 0$ for any $(p, q) \in E^C / \{(0, 1), (1, 0)\}$, i.e., $(p, q) \notin E \bigcup \{(0, 1), (1, 0)\}$.

Since each VNF can not be split, which implies it is exactly placed on an edge server or the cloud, we obtain

$$\sum_{k=0}^{M} x^k_{i,j} = 1, \qquad \forall 1 \leq i \leq m, 1 \leq j \leq n_i - 1. \tag{3}$$

Referring to the *Flow Conservation Law*, as for the data flow between VNF $F_{i,j}$ and $F_{i,j+1}$, we reach $\forall\, 0 \leq k \leq M$,

$$\sum_{p=0}^{M} w^{p,k}_{i,j} - \sum_{q=0}^{M} w^{k,q}_{i,j} = x^k_{i,j+1} - x^k_{i,j}. \tag{4}$$

The number of hops that data flow passes through at the edge network, from $F_{i,j}$ to $F_{i,j+1}$, is

$$z_{i,j} = \sum_{p=1}^{M} \sum_{q=1}^{M} w^{p,q}_{i,j}, \quad \forall 1 \leq i \leq m, 1 \leq j \leq n_i - 1. \tag{5}$$

We additionally define $z_{i,0} = 1$ and $z_{i,n_i} = 1$, showing at the edge, there is data flowing into the first VNF $F_{i,1}$ and flowing out of the last VNF F_{i,n_i} in SFC i.

Besides we define another *Boolean* variable, representing whether there is E&C communication latency between $F_{i,j}$ and $F_{i,j+1}$. As for any $1 \leq i \leq m, 1 \leq j \leq n_i - 1$,

$$Z_{i,j} = w^{1,0}_{i,j} + w^{0,1}_{i,j}. \tag{6}$$

Here we define $Z_{i,0} = x^0_{i,1}$ and $Z_{i,n_i} = x^0_{i,n_i}$, which implies if the first VNF of a SFC, $F_{i,1}$, is offloaded to the cloud, there still exists a transmitting data flow from edge to $F_{i,1}$ on the cloud. Similarly, if the last VNF of a SFC, F_{i,n_i}, is offloaded to the cloud, there is a transmitting data flow from F_{i,n_i} on cloud to the edge. This part is also considered in E&C communication latency.

In our model, there are below five optimization objectives.

- Resource cost at the edge $\mathbb{R}^{\mathbb{E}}$,
- Communication Latency between edge servers $\mathbb{L}^{\mathbb{E}}$,
- Resource cost on the cloud $\mathbb{R}^{\mathbb{C}}$,

- Communication Latency between edge and cloud $\mathbb{L}^{\mathbb{C}}$,
- The total processing latency $\mathbb{L}^{\mathbb{P}}$,

$$\mathbb{R}^{\mathbb{E}} = \sum_{k=1}^{M} y_k \cdot C_k, \qquad \mathbb{L}^{\mathbb{E}} = \sum_{i=1}^{m}\sum_{j=0}^{n_i} l_i \cdot z_{i,j},$$
$$\mathbb{R}^{\mathbb{C}} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} x^0_{i,j}, \cdot f_{i,j}, \quad \mathbb{L}^{\mathbb{C}} = \sum_{i=1}^{m}\sum_{j=0}^{n_i} L_i \cdot Z_{i,j}.$$
$$\mathbb{L}^{\mathbb{P}} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} x^0_{i,j} \cdot T_{i,j} + (1 - x^0_{i,j}) \cdot t_{i,j}$$

In all, the SFCD problem can be formulated as the below Integer Linear Programming (ILP) problem.

$$\min\ \alpha\mathbb{R}^{\mathbb{E}} + \beta\mathbb{L}^{\mathbb{E}} + \gamma\mathbb{R}^{\mathbb{C}} + \zeta\mathbb{L}^{\mathbb{C}} + \delta\mathbb{L}^{\mathbb{P}}$$
$$\text{s.t. } (1) - (6),$$

where $\alpha, \beta, \gamma, \zeta, \delta$ are weighting factors for adjusting the relative importance between objective components.

D. Problem Complexity

The SFCD problem has been proven to be NP-hard in many previous works like [13]. In short, it can be proved by reducing from the classical Bin Packing (BP) Problem, based on the resource cost alone. Besides, it can also be proved by the reduction from the Travelling Salesman Problem (TSP) if only the network latency is considered. It means the NP-hardness of SFCD comes from not only the resource cost part but also the latency part. Moreover, overall considering the edge-cloud continuum with five jointly optimization objectives makes the problem more complicated, posing challenges to our pursue of provable approximation algorithms.

IV. SFCD Limited at the Edge

In this section, we start from the edge-only cases of SFCD and design an approximation algorithm for it. In this edge-only case, obviously $\mathbb{R}^{\mathbb{C}} = \mathbb{L}^{\mathbb{C}} = 0$ and $\mathbb{L}^{\mathbb{P}} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{i,j}$, which is a given constant. Next, we prove competitive ratios of resource consumption $\mathbb{R}^{\mathbb{E}}$ and communication latency $\mathbb{L}^{\mathbb{E}}$ separately and finally get the approximation ratio of our algorithm by integrating these two parts.

A. Basic Ideas and Challenges

Acknowledged from the classical solutions for the BP problem, we find a suitable strategy for SFCD, called Next Fit strategy. The key idea of the Next Fit strategy for the BP problem is to sequentially pack items into bins, trying to place each item in the next available bin and moving to the next bin when necessary. It not only guarantees efficient resource utilization but also helps cut back communication latency by avoiding redundant data traffic. Additionally, in NF, there is no rule on sequences of bins and items, which gives us chances to design new rules specially for the SFC deployment. We plan to devise an approximation algorithm for SFCD, based on NF.

Algorithm 1: Double Spanning Tree (DST) Algorithm

Input: G and the servers $V_1, V_2, \cdots, V_M$.
Output: Sorted servers $V_{k_1}, V_{k_2}, \cdots, V_{k_M}$ and multi-hop paths T_j for the data flow between V_{k_j} and $V_{k_{j+1}}$.

Find the server V_p with maximal capacity and mark it.
Use DFS to obtain a spanning tree T with V_p as its root, where large-capacity nodes have higher priority.
Double T and delete a link between V_p and its child with the smallest capacity to get a path T'.
$j \leftarrow 1,\ k_j \leftarrow p$.
while $j < M$ **do**
    Find the next unmarked node of V_{k_j} on T', noted as V_q, and mark it.
    $j \leftarrow j + 1,\ k_j \leftarrow q$.
for $j = 1 \rightarrow M - 1$ **do**
    Build a sub-graph $G_j = (N_j, E_j)$ of G, where $N_j = \{V_{k_j}, V_{k_{j+1}}$, the nodes between them on path $T'\}$, $E_j \subseteq E$.
    Find the shortest path between V_{k_j} and $V_{k_{j+1}}$ in G_j with each edge weighted 1. This path is the multi-hop path T_j from V_{k_j} to $V_{k_{j+1}}$.

However, based on these ideas, challenges still exist. The first challenge is from the sophisticated network topology. The disconnection between some servers results in a multi-hop routing sub-problem between VNFs placed on different edge servers, which creates problems in mapping the virtual network of links between VNFs to the physical edge network with the bandwidth limits, i.e., virtual network embedding. To solve these troubles, we need to add a preparing sub-algorithm for virtual network embedding before applying the NF strategy. In particular, the task of this sub-algorithm is to optimize the number of hops between the adjacent employed servers by sorting servers. Moreover, different server capacities causes a gap between total used resources and the number of used servers, thus posing challenges to the proof of the approximation ratio on the communication latency.

B. Algorithm Design

As for the simplified SFCD limited at the edge, we devise an approximation algorithm called Chained Next Fit algorithm. In the beginning of CNF, we propose a preparing algorithm called Double Spanning Tree algorithm (Algorithm 1) to sort servers. In detail, we first choose a server with maximal capacity, noted as V_p. Then call the depth-first search (DFS) algorithm to obtain a spanning tree with V_p as the root. DFS is a graph traversal algorithm, whose fundamental concept is to explore as deeply as possible along a chosen branch before backtracking, ensuring a thorough exploration of the graph's structure. In the deep exploration process of DFS, each node may present multiple unexplored neighbor nodes and DFS does not provide a specific criterion for selection among these nodes. Thus, we design a capacity-degree sorting rule to cooperate

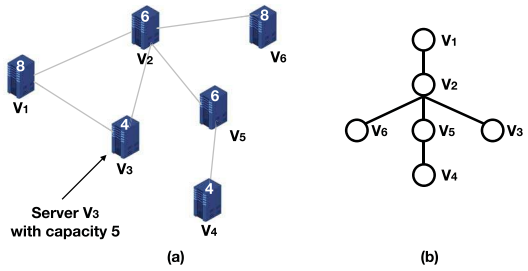


Fig. 2. A simple example for the DST algorithm.

with DFS as follows. During the deep exploration process of DFS, our algorithm consistently opts for unexplored neighbor nodes with the maximal server capacity. Take a simple network with 6 edge servers, as shown in Fig. 2(a), for example. We first set the maximal-capacity server V_1 as the root. Node V_1 has two unexplored neighbor nodes V_2 and V_3, where V_2 has a larger capacity, thus V_2 is explored after V_1. Next V_2 has three unexplored neighbor nodes V_2, V_5 and V_6, where V_6 has a larger capacity, thus V_6 is explored after V_2. Node V_6 does not have any neighbor nodes. Thus, it goes back to its parent node V_2, which has two unexplored neighbor nodes left. Among them, V_5 has a larger capacity. So explore V_5 next and then is its only one unexplored neighbor node V_4. Node V_4 has no neighbor nodes. Then, it goes back to its parent node V_5, which also has no unexplored neighbor nodes. Continue going back to its parent node V_2 and next explore its only one remaining unexplored neighbor node V_3. In all, we can obtain the spanning tree T as shown in Fig. 2(b).

Afterwards, we can get a path T' from this spanning tree T by doubling it and deleting a link between the root and its child. Starting from the root node of T and along the path T', we can get a traverse path of G by selecting all first-time appearing nodes. Finally, sort and reindex the servers in the traverse order.

Next, we sort and reindex SFCs in decreasing order of the corresponding one-hop communication latency l_i. After that, call the NF strategy to perform VNF placement. Specifically, it processes each VNF by following the chaining and sorted orderings of SFCs. As for each VNF, if the server capacity permits, place it on the currently considered server. If not, the VNF placement on the current server finishes, move to consider the next new edge server with enough capacity following the order of sorted servers, and place this VNF on this new server. Repeat the same procedures on the next VNF until all VNFs have been successfully placed on the edge servers or there is no enough edge resources for the VNF placement.

The detailed procedures of the CNF algorithm with **time complexity of** $O(N + mlog(m) + M^2)$ are shown in Algorithm 2.

It is worth to demonstrate that the solution produced by CNF is feasible under the assumption that each edge network connection has enough bandwidth for double of any data flow. Based on this premise, the NF strategy (line 5-16 in Algorithm 2) ensures the server capacity constraints in Ineq. 1 are satisfied. In CNF, we deal with VNFs in their chaining order (line 5 in Algorithm 2), which helps achieve the goal of avoiding redundant data traffic, because it guarantees there is at most one data flow between any two edge servers. Moreover, DST and the property of tree structure ensure that there are at most two hops (or data flows) passing through any edge network connection, which is explained in detail in Section IV-D of the conference version. So the connection bandwidth constraints in Ineq. (2) are satisfied.

Algorithm 2: Chained Next Fit (CNF) algorithm (Batched, Edge-only case)

Input: The list of VNFs $\{F_{i,j}\}$ and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$ and the number of used servers M'.

```
1  Sort and reindex servers by DST;
2  Sort and reindex SFCs so that l_1 ≥ l_2 ≥ ··· ≥ l_m;
3  k ← 1;
4  for i = 1 → m do
5      for j = 1 → n_i do
6          if V_k has enough capacity for F_{i,j} then
7              x^k_{i,j} = 1; (Place F_{i,j} on V_k)
8          else
9              while k < M do
10                 For all links (p, q) in the multi-hop path
                       T_k, let w^{p,q}_{i,j} = 1, k ← k + 1;
11                 if V_k has enough capacity for F_{i,j} then
12                     x^k_{i,j} = 1; (Place F_{i,j} on V_k)
13                     Break;
14                 else
15                     if k == M then
16                         Return 0; (Fail)
17 M' ← k.
```

C. Bound of Resource Part

Below, we show CNF has an asymptotic approximation ratio of 2 to the optimal solution (OPT) on resource cost $\mathbb{R}^{\mathbb{E}}$. For proof of the ratio, we use functions of $C(\cdot)$ and $c(\cdot)$ to respectively represent the capacity of an edge server and the occupied part. Obviously, for any $1 \le k \le M'$, $c(V_k) \le C(V_k)$.

Besides we assume $V_1, \cdots, V_{M'}$ are used servers by CNF while $V_1^*, \cdots, V_{M^*}^*$ are those by OPT. Since they both deal with the same VNF set $\{F_{i,j}\}$,

$$\sum_{k=1}^{M'} c(V_k) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{i,j} = \sum_{k=1}^{M^*} c(V_k^*).$$

Theorem 1:

$$\mathbb{R}^{\mathbb{E}}{}_{CNF} < 2 \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C,$$

where $C = \max C_k$.

Proof: Omitted, please refer to the conference version [25], where the proof of this theorem is standalone. □

With more information on sizes of VNFs and servers, we can give CNF a better approximation ratio. Here, we define a related parameter r.

$$r = \frac{\min C_k}{\max f_{i,j}}. \tag{7}$$

Theorem 2: When $r > 1$, $\mathbb{R}^{\mathbb{E}}{}_{CNF} < \frac{r}{r-1} \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C$, where $C = \max C_k$ is constant.

Proof: For any $1 \le k < M'$, the first VNF placed on V_{k+1} cannot be placed onto V_k. Denote its size as f^1_{k+1}, then we have

$$c(V_k) + f^1_{k+1} > C_k.$$

By the definition of r, we know that for $1 \le k < M'$,

$$f^1_{k+1} \le \max_{1 \le p \le N} \{f_p\} = \frac{1}{r} \min_{1 \le q \le M} \{C_q\} \le \frac{1}{r} C_k. \tag{8}$$

Then

$$c(V_k) > C_j - f^1_{k+1} \ge C_k - \frac{1}{r} C_k = \frac{r-1}{r} C(V_k).$$

When $R > 1$,

$$C(V_k) < \frac{r}{r-1} c(V_k), \quad \forall 1 \le k < M'.$$

Therefore,

$$\begin{aligned}
\mathbb{R}^{\mathbb{E}}{}_{CNF} &= \sum_{k=1}^{M'} C(V_j) = \sum_{k=1}^{M'-1} C(V_j) + C(V_{M'}), \\
&< \frac{r}{r-1} \sum_{r=1}^{M'} c(V_k) + C = \frac{r}{r-1} \sum_{k=1}^{M^*} c(V^*_k) + C, \\
&\le \frac{r}{r-1} \sum_{j=1}^{M^*} C(V^*_k) + C = \frac{r}{r-1} \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C.
\end{aligned}$$

□

Obviously, when $r > 2$, $\frac{r}{r-1} < 2$, which implies in this special case, we can give a smaller approximation ratio of $\frac{r}{r-1}$ to SF.

D. Bound of Communication Part

Here we show CNF also has a constant ratio to OPT on $\mathbb{L}^{\mathbb{E}}$.

Theorem 3:

$$\mathbb{L}^{\mathbb{E}}{}_{CNF} \le 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}.$$

Proof: Omitted, please refer to the conference version [25], where the proof of this theorem is standalone. □

E. Approximation Ratio of CNF in the Edge-Only Case

Then we can combine upper bounds of $\mathbb{R}^{\mathbb{E}}{}_{CNF}$ and $\mathbb{L}^{\mathbb{E}}{}_{CNF}$ to prove the constant approximation ratio of CNF in SFCD. Denote by CNF and OPT the total cost respectively by CNF and OPT. Combined with Theorem 1, 3, we reach the below theorem.

Theorem 4: $CNF \le 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot OPT + C$, where $C = \alpha \cdot \max C_k$.

V. SFCD for Edge-Cloud Continuum

As for some special SFCs which can rely on the powerful computing resource of the cloud, like GPU or FPGA, to reduce processing latency, if such acceleration advantages can offset high communication latency, i.e., $\delta \sum_{j=1}^{n_i} (t_{i,j} - T_{i,j}) \ge 2\zeta \cdot L_i$, obviously these SFCs had better be offloaded to the cloud for lower processing latency. Otherwise, we prioritize edge resources due to their low communication latency.

To maximize the exploitation of edge resources, it is a natural idea to prioritize VNFs at the edge as resources allow and then offload all the remaining VNFs onto the cloud. Some existing related work [4], [5] just did so. However, this approach has a drawback in that it fails to proactively determine which SFCs to be offloaded to the cloud. Offloading various SFCs to the cloud can have a substantial impact on the overall communication latency. Thus, such a trivial approach may cause increased communication latency, especially the outside-edge communication latency, and is not a good option. In this section, we plan to explore new algorithms for the scenarios of the edge-cloud continuum, based on the CNF algorithm for the edge-only cases. The critical challenge is how to control the SFCs (or VNFs) offloaded to the cloud to minimize the employed cloud resources and the produced outside-edge communication latency.

A. Algorithm Design

Below we will design an approximation algorithm called Edge-first Chained Next Fit algorithm (ECNF). Here we define a new latency parameter $\widetilde{L_i} = L_i - \frac{\delta}{2\zeta} \sum_{j=1}^{n_i} (t_{i,j} - T_{i,j})$. If $\widetilde{L_i} \le 0$, offload the whole SFC i to the cloud (line 1 in Algorithm 3). Below we focus on the remaining SFCs with $\widetilde{L_i} > 0$.

From the above section, we can see CNF does a good job on the edge-only cases, which is also a special case belonging to the edge-cloud continuum. Thus, as for such cases, just call CNF to deploy SFCs at the edge (line 2-4 in Algorithm 3). Below we focus on the cases that CNF can NOT successfully deploy all SFCs at the edge. That is to say, some VNFs or SFCs are needed to be offloaded to the cloud.

According to Theorem 1, CNF can work on maximizing the exploitation of edge resources and minimizing the needed cloud resources for the remaining VNFs. But it does not involve the control of what the SFCs (or VNFs) to be offloaded to the cloud to minimize the additional network latency, i.e., the produced outside-edge communication latency minus the cloud processing acceleration. It is our new task in this section.

In our proposal, we formulate the following basic rule, which can help reduce the outside-edge communication latency. We always offload the whole SFC i or a continuously chaining segment of SFC i to the cloud. In this case, the additional network latency for SFC i is $2\zeta \cdot L_i - \delta \sum_{j=1}^{n_i} (t_{i,j} - T_{i,j}) = 2\zeta \cdot \widetilde{L_i}$. Fortunately, the remaining VNFs after CNF must conform to this rule.

Besides, in order to proactively determine which SFCs to be offloaded to the cloud, we consider an SFC selection problem

Algorithm 3: Edge-first Chained Next Fit (ECNF) algorithm (edge-cloud continuum)

Input: The list of VNFs $\{F_{i,j}\}$ and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$.

1 Offload all SFCs with $\widetilde{L_i} \leq 0$ to the cloud and reindex the remaining SFCs from 1 to m' ;
2 Call default CNF to place the remaining SFCs;
3 **if** *success* **then**
4 END! (Edge-only cases)
5 Sort and reindex SFCs so that $\frac{\widetilde{L}_1}{F_1} \geq \frac{\widetilde{L}_2}{F_2} \geq \cdots \geq \frac{\widetilde{L}_{m'}}{F_{m'}}$;
6 $\widetilde{C} \leftarrow \sum_{k=1}^{M} C_k$, $Sum \leftarrow F_1$, $i \leftarrow 1$;
7 **while** $Sum \leq \frac{1}{2}\widetilde{C}$ **do**
8 $i \leftarrow i+1$, $Sum \leftarrow Sum + F_i$;
9 Call CNF with DST-revised (Algorithm 4) to place SFCs 1 to $i-1$ at the edge;
10 **while** *there exist edge resources left* **do**
11 Call CNF with DST-revised (Algorithm 4) to deal with SFC i;
12 $i_0 \leftarrow i$, $i \leftarrow i+1$;
13 Offload the rest VNFs of SFC i_0, as well as all remaining SFCs, from i_0+1 to m', to the cloud.

for the cloud. Assume the total needed cloud resource is R_c. Note that Theorem 1 ensures if $R_c \geq \frac{1}{2}\sum_{k=1}^{M} C_k$, CNF can always successfully place all unselected SFCs at the edge. Then the SFC selection problem for the cloud with the objective of minimizing the additional network latency can be formulated as a below 0-1 min-knapsack problem.

$$\min \quad \sum_{i=1}^{m} 2\zeta\widetilde{L_i} \cdot X_i,$$
$$\text{s.t.} \quad \sum_{i=1}^{m} F_i \cdot X_i \geq R_c,$$

where X_i is a Boolean variable, representing if SFC i is offloaded to cloud or not; $F_i = \sum_{j=1}^{n_i} f_{i,j}$ denotes the total size of all VNFs in SFC i, also represents the size of SFC i.

This 0-1 min-knapsack problem can be dealt with by a greedy solution. Sort and reindex SFCs by $\frac{\widetilde{L}_i}{F_i}$ in increasing order so that

$$\frac{\widetilde{L}_1}{F_1} \leq \frac{\widetilde{L}_2}{F_2} \leq \cdots \leq \frac{\widetilde{L}_m}{F_m}.$$

Then keep choosing SFCs following index order until the total sizes of all choice SFCs reach R_c. Combined with the purpose of maximizing the exploitation of edge resources, we sort and reindex SFCs in the inverse order (line 5 in Algorithm 3) and deploy SFCs at the edge in order as much as possible.

CNF can work on deploying SFCs at the edge and maximize the exploitation of edge resources. But the reindexing step of SFCs in line 2 of Algorithm 2 conflicts with the above sorting step by $\frac{\widetilde{L}_i}{F_i}$. Thus when employing CNF, we must ensure enough

Algorithm 4: DST-revised (for edge-cloud continuum)

Input: The list of VNFs $\{F_{i,j}\}$ and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$.

1 Start from the server $V_p (p=0)$ and mark it;
2 Use DFS to obtain a spanning tree T with V_p as its root, where small-capacity nodes have higher priority;
3 Call line 3-7 of DST (Algorithm 1);
4 Sorting and reindex server in the inverse order;
5 Call line 8-10 of DST (Algorithm 1).

resources at the edge. Otherwise, the left SFCs are not necessarily those with with small $\frac{\widetilde{L}_i}{F_i}$, which may cause high out-side edge communication latency. Above all, we first call CNF with the input of the most i_0 smallest-index SFCs satisfying the total sizes are no more than $\frac{1}{2}\sum_{k=1}^{M} C_k \overset{\text{def}}{=} \frac{1}{2}\widetilde{C}$, i.e., $\sum_{i=1}^{i_0} F_i \leq \frac{1}{2}\widetilde{C}$ (line 6-9 in Algorithm 3), because Theorem 1 tells us there is always enough edge servers for these SFCs. Then as for the rest SFCs, we can not ensure the adequate edge resources for them, thus in order to avoid line 2 of Algorithm 2 invalidating the order of SFCs, we had better limit the input with only one SFC when calling CNF. Hence, keep calling CNF to deal with the next SFC following the index order, until there is no enough edge resources and CNF breaks (line 10-12 in Algorithm 3). Finally, we offload the rest VNFs of the current SFC i0, as well as all remaining SFCs, from i_0+1 to m', to the cloud (line 13 in Algorithm 3). The detailed ECNF algorithm with **time complexity of** $O(N + mlog(m) + M^2)$ is shown in Algorithm 3.

It is worth noticing that in CNF, we sort servers by DST (Algorithm 1) where the first node is special. However, according to the above algorithm steps, here we want the last server to be special, being the central server V_0. It is because in ECNF, the last server is the only one which must communicate with the cloud and the central server V_0 the one nearest to the cloud among all servers at the edge. Thus here we simply revise DST as Algorithm 4 to make the last server the central server after sorting. In ECNF except the edge-only cases, i.e., in line 9, 11 of Algorithm 3, we always substitute DST by this revised DST when calling CNF.

Then we will illustrate the deployment scheme gained by ECNF is feasible. First CNF guarantees all constraints at the edge, which we has analyzed in the above section. Besides, ECNF combined with the revised DST algorithm ensures only the central server needs to communicate with the cloud, which conforms to the connection bandwidth constraints between edge and cloud, i.e., $p=0$ or $q=0$ in Inequality (2).

B. Approximation Ratio of ECNF in the Edge-Cloud Continuum

We now prove ECNF has constant approximation ratios to OPT for two different cases. In proof, denote by $ECNF$ and OPT the total cost of ECNF and that of OPT. Besides, denote the total size of all VNFs with $\widetilde{L_i} > 0$ as $\widetilde{F} = \sum_{i=1}^{m'} F_i$.

Theorem 5: $ECNF \le r \cdot OPT + C$, where if $\widetilde{F} \le \frac{1}{2}\widetilde{C}$,

$$C = \alpha \cdot \max C_k \text{ and } r = 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil;$$

If $\widetilde{F} > \frac{1}{2}\widetilde{C}$,

$$C = 2\zeta \cdot \max \widetilde{L}_i \text{ and } r = \max \left\{ 4 \left\lceil \frac{\max C_k}{\min C_k} \right\rceil, \frac{2\alpha\widetilde{C} + (2\widetilde{F} - \widetilde{C})\left(\gamma + 2\zeta \max_i \frac{\widetilde{L}_i}{F_i}\right)}{2\alpha \min\left\{\widetilde{F}, \widetilde{C}\right\} + 2\max\left\{\widetilde{F} - \widetilde{C}, 0\right\}\left(\gamma + 2\zeta \min_i \frac{\widetilde{L}_i}{F_i}\right)} \right\}. \tag{9}$$

And when $\widetilde{F} \to \infty$, $r = \max\left\{4\left\lceil \frac{\max C_k}{\min C_k}\right\rceil, \frac{\gamma + 2\zeta \max_i \frac{\widetilde{L}_i}{F_i}}{\gamma + 2\zeta \min_i \frac{\widetilde{L}_i}{F_i}}\right\}$ is constant. When $\alpha = 0$, $\frac{1}{2}\widetilde{C} < \widetilde{F} \le \widetilde{C}$, r has another format, $r = 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil + \frac{\max C_k}{\min l_i} \cdot \frac{(2\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max_i \frac{\widetilde{L}_i}{F_i})}{\beta(\widetilde{F} + \widetilde{C})}$.

Proof: First, as for the SFCs with $\widetilde{L_i} \le 0$, offloading the entire chain to the cloud is the optimal choice. Below, our analysis focuses on the deployment of the rest SFCs.

When $\widetilde{F} \le \frac{1}{2}\widetilde{C}$, ECNF is reduced to CNF and SFCD is limited at the edge. Referring to Theorem 4,

$$r = 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \text{ and } C = \alpha \cdot \max C_k.$$

When $\widetilde{F} > \frac{1}{2}\widetilde{C}$, we analyse as below. First, it is easy to get

$$\mathbb{R}^{\mathbb{E}}{}_{e.f} \le \widetilde{C}, \quad \mathbb{R}^{\mathbb{E}}{}_{OPT} \ge \min\left\{\widetilde{F}, \widetilde{C}\right\}.$$

Since ECNF combined with the revised DST ensures only the central server needs to communicate with the cloud, $\mathbb{L}^{\mathbb{E}}{}_{e.f}$ comes totally from the data flow between VNFs placed at the edge, like the green solid line in Fig. 1, rather than the inside-edge part of the communication data flow between edge and cloud like the red solid line in Fig. 1. Based on this fact and Theorem 3, we can obtain $\mathbb{L}^{\mathbb{E}}{}_{e.f} \le 4\left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot OPT_c$, where OPT_c means the minimal latency when placing these VNFs at the edge. It is the latency of optimal solutions for a new problem with the task of placing these VNFs at the edge and the goal of only minimizing the communication latency. In OPT, the optimal solution for the whole SFCD problem, there are more VNFs placed at the edge because of the low latency of edge computing. Thus, $\mathbb{L}^{\mathbb{E}}{}_{OPT} \ge OPT_c$.

Line 10-12 in Algorithm 3 guarantees all edge servers are occupied, hence combined with Theorem 1, we can reach the conclusion that the total sizes of VNFs placed at the edge $\ge \frac{1}{2}\widetilde{C}$, which implies $\mathbb{R}^{\mathbb{C}}{}_{e.f} \le \widetilde{F} - \frac{1}{2}\widetilde{C}$.

In OPT, the total sizes of VNFs placed at the edge $\le \widetilde{C}$, thus if $\widetilde{F} \ge \widetilde{C}$, $\mathbb{R}^{\mathbb{C}}{}_{OPT} \ge \widetilde{F} - \widetilde{C}$. If $\widetilde{F} < \widetilde{C}$, there exists a possibility that all SFCs are placed at the edge, so in this case, we can only get $\mathbb{R}^{\mathbb{C}}{}_{OPT} \ge 0$. In all, $\mathbb{R}^{\mathbb{C}}{}_{OPT} \ge \max\left\{\widetilde{F} - \widetilde{C}, 0\right\}$.

In ECNF, SFCs from 1 to $i_0 - 1$ are successfully placed on edge servers while SFCs i_0 to m' are totally or partially offloaded to the cloud. Thus, $\zeta\mathbb{L}^{\mathbb{C}}{}_{e.f} + \delta\mathbb{L}^{\mathbb{P}}{}_{e.f} = \zeta\sum_{i=i_0}^{m'} 2\widetilde{L_i} + \delta\sum_{i=1}^{i_0 1}\sum_{j=1}^{n_i} t_{i,j} + \delta\sum_{i=i_0}^{m'}\sum_{j=1}^{n_i} T_{i,j} = \zeta\sum_{i=i_0}^{m'} 2\widetilde{L_i} + \delta\sum_{i=1}^{m'}\sum_{j=1}^{n_i} t_{i,j} \le 2\zeta\widetilde{L_{i_0}} + 2\zeta(\max\frac{\widetilde{L}_i}{F_i})\sum_{i=i_0+1}^{m} F_i + \delta \cdot t_c$, where $\sum_{i=1}^{m'}\sum_{j=1}^{n_i} t_{i,j}$, noted as t_c, is a fixed constant related to the given information of SFCs, which is independent of the SFC deployment schemes. Besides, Theorem 1 demonstrates $\sum_{i=1}^{i_0} F_i \ge \frac{1}{2}\widetilde{C}$, which implies $\sum_{i=i_0+1}^{m} F_i = F - \sum_{i=1}^{i_0} F_i \le F - \frac{1}{2}C$. Therefore, $\zeta\mathbb{L}^{\mathbb{C}}{}_{e.f} + \delta\mathbb{L}^{\mathbb{P}}{}_{e.f} \le 2\zeta \max \widetilde{L}_i + \zeta \max \frac{\widetilde{L}_i}{F_i} \cdot (2\widetilde{F} - \widetilde{C}) + \delta \cdot t_c$.

When it comes to OPT, if some VNFs in SFC i are offloaded to cloud, the outside-edge communication latency of this chain is at least $2\widetilde{L}_i$. And the total size of SFCs that contain VNFs offloaded to cloud is larger than or equal to the used cloud resources $\mathbb{R}^{\mathbb{C}}{}_{OPT}$. Thus, $\zeta\mathbb{L}^{\mathbb{C}}{}_{opt} + \delta\mathbb{L}^{\mathbb{P}}{}_{opt} \ge \min\frac{2\widetilde{L_i}}{F_i} \cdot \max\left\{\widetilde{F} - \widetilde{C}, 0\right\} + \delta \cdot t_c$.

Above all,

$$\begin{aligned} ECNF &= \alpha\mathbb{R}^{\mathbb{E}}{}_{e.f} + \beta\mathbb{L}^{\mathbb{E}}{}_{e.f} + \gamma\mathbb{R}^{\mathbb{C}}{}_{e.f} + \zeta\mathbb{L}^{\mathbb{C}}{}_{e.f} + \delta\mathbb{L}^{\mathbb{P}}{}_{e.f}, \\ &\le \alpha\widetilde{C} + 4\left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot (\beta \cdot OPT_c) + 2\zeta \cdot \max \widetilde{L_i}, \\ &\quad + \left(\frac{1}{2}\gamma + \zeta \cdot \max \frac{\widetilde{L}_i}{f_i}\right) \cdot (2\widetilde{F} - \widetilde{C}) + \zeta \cdot t_c, \\ OPT &= \alpha\mathbb{R}^{\mathbb{E}}{}_{opt} + \beta\mathbb{L}^{\mathbb{E}}{}_{opt} + \gamma\mathbb{R}^{\mathbb{C}}{}_{opt} + \beta\mathbb{L}^{\mathbb{C}}{}_{opt} + \delta\mathbb{L}^{\mathbb{P}}{}_{opt}, \\ &\ge \alpha \min\left\{\widetilde{F}, \widetilde{C}\right\} + \beta \cdot OPT_c \\ &\quad + \left(\gamma + 2\zeta \cdot \min \frac{\widetilde{L}_i}{f_i}\right) \cdot \max\left\{\widetilde{F} - \widetilde{C}, 0\right\} + \zeta \cdot t_c. \end{aligned}$$

Since $\frac{a+b+e}{c+d+e} \le \max\left\{\frac{a}{c}, \frac{b}{d}, 1\right\}$, letting $b = 4\left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot (\beta \cdot OPT_c)$, $d = \beta \cdot OPT_c$, and $e = \zeta \cdot t_c$ we can get Eq. 9.

Since there is at least one hop between two employed edge servers for SFC i, $OPT_c \ge \sum_{i=1}^{m}\left(\frac{F_i}{\max C_k} + 1\right) \cdot l_i \ge (\frac{\widetilde{F}}{\max C_k} + m) \cdot \min l_i \ge \frac{\min l_i}{\max C_k}(\widetilde{F} + \widetilde{C})$. Thus, when $\alpha = 0$, $\frac{1}{2}\widetilde{C} < \widetilde{F} \le \widetilde{C}$, r has another format. $r = 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil + \frac{\max C_k}{\min l_i} \cdot \frac{(4\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max_i \frac{\widetilde{L}_i}{F_i})}{\beta(\widetilde{F} + \widetilde{C})}$. □

VI. The Online SFCD

In above discussion, we assume that the information of all the SFCs is known before the deployment. Here we extend the SFCD problem to an online version, where the SFCs arrive sequentially in an online manner. At each time slot, we do not know the information of SFCs that will arrive in the future and we only know the information of SFCs that have arrived in the past. If we start to make SFC deployment and flow routing only after all of them have arrived, the earlier SFCs will suffer high delays. Therefore, we provide an online algorithm, called the online chained next fit (OCNF) algorithm, to solve the online SFCD problem by deploying SFCs and routing the flow after each new SFC arrives.

A. Basic Ideas and Challenges

We want to design OCNF based on the framework of ECNF so that the performance guarantee of algorithms on resource

cost and network latency can be maintained as much as possible. Luckily, it is easy to keep the approximation ratio on resource cost, i.e., Theorem 1, Theorem 2, because the design of the resource-related algorithms is based on the NF strategy, which itself is an online algorithm and thus can fit the online mechanism successfully. However, the design of the latency-related algorithms is a totally different story. Line 5 of ECNF and line 2 of CNF, which is called in ECNF in lines 2 and 9, both involve the sorting of latency-related coefficients, e.g. $l_i, \widetilde{L_i}$, which only works when the information of all SFCs is known. Thus, these two-line codes can NOT be adopted in OCNF. Line 5 of ECNF is only employed to obtain better average performance but does not make a difference in the worst-case performance in Theorem 4. The trouble is that line 2 of CNF (Algorithm 2) is a key step for the proof of the latency approximation ratio (Seen in the proof of Theorem 3 in the conference version [25]). Thus, the key challenge in OCNF design is how to achieve a latency approximation ratio like Theorem 3.

B. Algorithm Design

Since the online mechanism affects the information of SFCs, we consider doing more jobs on the network topologies. Specifically, in CNF, we sort servers by DST. *Can we get a better traverse path on the network graph so that the maximal number of hops between two adjacent nodes in the traverse path is limited?* If so, we can achieve a latency approximation ratio via steps like Eq. 10.

Such a problem is equivalent to a Hamiltonian cycle (/path) problem. In detail, we consider a new graph G' with the same vertex set V as the network graph G and each pair of vertices is connected by an edge in G' if and only if they are joined by a path with at most d edges (or hops) in G. Then the problem of finding a traverse path on the network graph G so that the maximal number of hops between two adjacent nodes in the path is equivalent to the problem of finding a Hamiltonian cycle (/path) in the new graph G'. In terms of such a Hamiltonian cycle problem, H. Fleischner [26] has proved an existence theorem in 1974 that the square[2] of every two-connected graph[3] is Hamiltonian, i.e., we can always find a Hamiltonian cycle in the square of every two-connected graph. Then H. T. Lau [27] proposed an $O(|V|^2)$ algorithm in 1980 for producing a Hamiltonian cycle in the square of a 2-connected graph.

Based on Fleischner's theorem and Lau's algorithm, below we first devise a new preparing algorithm for OCNF, called the Hamiltonian path traverse (HPT) algorithm. It can substitute DST to sort servers, i.e., find a Hamiltonian path of G' to traverse the network graph G and sort the servers following this traverse path. Specifically, we first use the depth-first search (DFS) algorithm to obtain a spanning tree T. Then based on T, we can get a new graph T' by the square of graph T, i.e., $T' = T^2$. By the definition of T', we can claim that T' must be a two-connected graph. Let G' be the square of graph T'. According to Fleischner's theorem and Lau's algorithm, we can find a Hamiltonian cycle in G'. Then, it is easy to obtain a Hamiltonian path from a Hamiltonian cycle by deleting an edge with V_1 as an endpoint. Such a Hamiltonian path is exactly the traverse path we obtained and we can sort the servers following this path from another endpoint to V_1.

Algorithm 5: Hamiltonian path traverse (HPT) Algorithm

Input: The physical network $G = (V, E)$, where $V = \{V_1, \cdots, V_M\}$.
Output: The sorted servers $V_{k_1}, V_{k_2}, \cdots, V_{k_M}$ and multi-hops paths T_j for the data flow between V_{k_j} and $V_{k_{j+1}}$.

1 Call DFS to get a spanning tree T of graph G;
2 Get a new graph T' by the square of T;
3 Get a new graph G' by the suqare of T';
4 Call Lau's $O(|V|^2)$ algorithm to produce a Hamiltonian cycle H^C in G';
5 Obtain a Hamiltonian path H^P from the Hamiltonian cycle H^P by deleting an edge with V_1 as an endpoint.
6 Note the other endpoint of path H^P except V_1 as V_p;
7 $j \leftarrow 1,\ k_j \leftarrow p$;
8 **while** $j < M$ **do**
9 Find the next node of V_{k_j} on H^P, noted as V_q;
10 $j \leftarrow j + 1,\ k_j \leftarrow q$;
11 **for** $j = 1 \rightarrow M - 1$ **do**
12 Find the shortest path between V_{k_j} and $V_{k_{j+1}}$ in G with each edge weighted 1, noted as T_j .

The detailed procedures of HPT with the **time complexity of** $O(M^2)$ are shown in Algorithm 5.

After sorting the servers by HPT, we start to deploy SFC i in the edge-cloud continuum. If the currently considered server is the cloud, just put all the VNFs of SFC i on the cloud. Otherwise, we employ the NF strategy to do deployment for the newly arrived SFC i following the chained order of VNFs. Specifically, if a VNF fits inside the currently considered edge server, it is placed on this server. Otherwise, the placement on the current server ends, move to consider the next new edge server with enough capacity following the order of sorted server, and place the current VNF on this new server Repeat the same procedures on the next VNF until all VNFs are placed on the edge servers or there is not enough resource at the edge. If there is not enough resource at the edge, put all the rest VNFs of SFC i on the cloud.

The detailed OCNF algorithm with **time complexity of** $O(N + M^2)$ is shown in Algorithm 6.

C. Approxi. Ratio of OCNF in the Edge-Only Case

Theorem 6: $\mathbb{R}^{\mathbb{E}}{}_{OCNF} \leq \min\{2, \frac{r}{r-1}\} \cdot \mathbb{R}^{\mathbb{E}}{}_{OCNF} + C$, where $C = \max C_k$ and $r = \frac{\min C_k}{\max f_{i,j}}$.

[2]The square of a graph G, i.e., G^2, is a graph with the same set of vertices of G and each pair of vertices is connected by an edge in G^2 if and only if they are joined by a path with at most 2 edges in G.

[3]if any one vertex were to be removed, the graph will remain connected.

Algorithm 6: Online Chained Next Fit (OCNF) algorithm (Online, edge-cloud continuum)

Input: The new arrived SFC i, the current considered server V_k, and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$.

```
Sort and reindex servers by HPT;
if k > 0 (there is resource left at the edge) and L̃_i > 0 then
    for j = 1 → n_i do
        if V_k has enough capacity for F_{i,j} then
            x^k_{i,j} = 1; (Place F_{i,j} on V_k)
        else
            while k < M do
                For all links (p, q) in the multi-hop path T_k, let w^{p,q}_{i,j} = 1, k ← k + 1;
                if V_k has enough capacity for F_{i,j} then
                    x^k_{i,j} = 1; (Place F_{i,j} on V_k)
                    Break;
            if k = M and x^M_{i,j} = 0 then
                Put the rest VNFs of SFC i on the cloud; (x^0_{i,j} = 1)
                k = 0;
                break;
else
    Put all VNFs of SFC i on the cloud; (x^0_{i,j} = 1)
```

Proof: OCNF stills employ the NF strategy when placing VNFs on the edge, so we can obtain the conclusion the same as Theorem 1 and Theorem 2. □

Theorem 7: $\mathbb{L}^{\mathbb{E}}{}_{OCNF} < 8 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}$.

Proof: According to HPT, the. number of hops between two adjacent sorted servers is at most $2 * 2 = 4$, thus

$$\sum_{j=0}^{n_i} z_{i,j} = 2 + \sum_{j=1}^{n_i - 1} z_{i,j} \leq 2 + 4 * (M_i - 1) < 4 * M_i, \quad (10)$$

where M_i represents the number of employed servers when placing SFC i by OCNF.

Similarly, we use M_i^* to represent the number of used servers when placing SFC i by OPT. Additionally, denote by V_{p_i} and V_{q_i} the first used server of SFC i by CNF and the last one. According to these definitions, we can obtain

$$M_i^* \geq \left\lceil \frac{\sum_{j=1}^{n_i} f_{i,j}}{\max\limits_{1 \leq k \leq M} C_k} \right\rceil \quad \text{and} \quad M_i \leq \left\lceil \frac{\sum_{k=p_i}^{q_i} C(V_k)}{\min\limits_{1 \leq k \leq M} C_k} \right\rceil.$$

In OPT, there is at least one hop between two employed edge servers in SFC i. After adding one hop for data flowing in and one for data flowing out, we reach

$$z_i^* \geq M_i^* + 1.$$

In sum,

$$M_i \leq \left\lceil \frac{\sum_{k=p_i+1}^{q_i - 1} C(V_k) + C(V_{p_i}) + C(V_{q_i})}{\min C_k} \right\rceil, \quad (11)$$

$$\leq \left\lceil \frac{2 \sum_{j=1}^{n_i} f_{i,j} + 2 \max C_k}{\min C_k} \right\rceil, \quad (12)$$

$$\leq 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot M_i^* + 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil,$$

$$\leq 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot z_i^*,$$

where $\lceil \cdot \rceil$ is the ceiling function, Format 11$\leq$ Format 12 stems from Theorem 1.

Referring to Ineq. 10, we can get the conclusion that

$$\mathbb{L}^{\mathbb{E}}{}_{OCNF} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} z_{i,j} \cdot l_i < 4 \cdot \sum_{i=1}^{m} M_i \cdot l_i,$$

$$\leq 4 \cdot \sum_{i=1}^{m} 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot z_i^* \cdot l_i,$$

$$= 8 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}.$$

□

Theorem 8: $OCNF < 8 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot OPT + C$, where $C = \max C_k$.

D. Approximation Ratio of OCNF in the Edge-Cloud Continuum

Similar as the proof of Theorem 5 and combined with Theorem 6, 7, we get

Theorem 9: $OCNF \leq r \cdot OPT + C$, where if $\widetilde{F} \leq \frac{1}{2}\widetilde{C}$,

$$C = \alpha \cdot \max C_k \text{ and } r = 8 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil;$$

If $\widetilde{F} > \frac{1}{2}\widetilde{C}$,

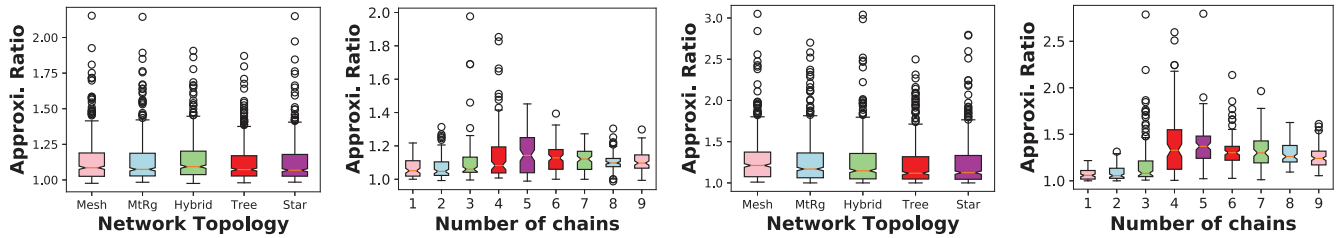
$$C = 2\zeta \cdot \max \widetilde{L_i} \text{ and } r = \max \left\{ 8 \left\lceil \frac{\max C_k}{\min C_k} \right\rceil, \right.$$

$$\left. \frac{2\alpha\widetilde{C} + (2\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max\limits_i \frac{\widetilde{L}_i}{F_i})}{2\alpha \min\left\{\widetilde{F}, \widetilde{C}\right\} + 2\max\left\{\widetilde{F} - \widetilde{C}, 0\right\}(\gamma + 2\zeta \min\limits_i \frac{\widetilde{L}_i}{F_i})} \right\}. \quad (13)$$

And when $\widetilde{F} \to \infty$, $r = \max \left\{ 8 \left\lceil \frac{\max C_k}{\min C_k} \right\rceil, \frac{\gamma + 2\zeta \max\limits_i \frac{\widetilde{L}_i}{F_i}}{\gamma + 2\zeta \min\limits_i \frac{\widetilde{L}_i}{F_i}} \right\}$ is constant. When $\alpha = 0$, $\frac{1}{2}\widetilde{C} < \widetilde{F} \leq \widetilde{C}$, r has another format, $r = 8 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil + \frac{\max C_k}{\min l_i} \cdot \frac{(2\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max\limits_i \frac{\widetilde{L}_i}{F_i})}{\beta(\widetilde{F} + \widetilde{C})}$.

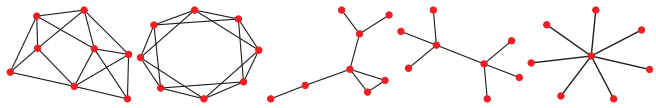
VII. Performance Evaluation

In this section, we perform extensive simulations on different network topologies to compare the performance of ECNF and OCNF with optimal solutions and benchmarks.



(a) ECNF on Diff. Topologies (b) ECNF on Star Topology (c) OCNF on Diff. Topologies (d) OCNF on Star Typology

Fig. 3. Approximation ratios of ECNF and OCNF on topologies with 8 nodes.



(a) Mesh (b) MtRg (c) Hybrid (d) Tree (e) Star

Fig. 4. 5 different network topologies with 8 nodes.

A. Simulation Setup

The evaluation of the proposed algorithms in different scenarios is performed through simulations. At the time we write, several standard simulation frameworks for the edge have been proposed, such as iFogSim [28] and FogBus [29]. However, they all do not suit our model. Specifically, iFogSim does not support device-to-device communication, and FogBus does not consider the communication between tasks in their model, while the communication between the adjacent chained VNFs is an important consideration in the model of SFCD. Compared with them, the Monte-Carlo simulations are more suited for the variability of the targeted environment, according to [30]. Thus, we adopt a Monte Carlo simulator and add our model to the simulator framework.

As for the infrastructure of the model, we set the one-hop communication latency of an SFC i, l_i, randomly ranging from 0.5 ms to 1 ms, and the edge-and-cloud communication latency of an SFC, L_i, randomly ranging from 10 ms to 50 ms. Besides, we randomly set the processing latency of each VNF on edge server $t_{i,j}$, with a 90% likelihood of ranging from 0.5 ms to 1 ms and a 10% likelihood of ranging from 1 ms to 100 ms. We configure the VNF processing latency on the cloud to randomly range from 0.3 to 1 times that on edge servers. Additionally, we set the bandwidth capacity of each hop in the edge network, $B_{p,q}$, as 1300 Mbps (the bandwidth of Wireless 802.11ac). The flow rate of SFC i, b_i, ranges from 0.5 Mbps to 5 Mbps. Additionally, we set $f_{i,j} \sim \mathbb{N}(2, 0.5)$.

In below simulations, as for each group setting, we always conduct 100 groups of simulations to show average cases.

B. Simulations on 5 Small-Scale Random Network Topologies (Performance Comparisons With OPTs)

1) Network Topology and Server Capacity Setup: Since SFCD is NP-hard, the time cost of traversal search for OPT grows exponentially as the problem scale, i.e., N, M, increases. Thus, we first perform simulations on some representative small-scale network topologies with 8 nodes shown in Fig. 4. We call these topologies the mesh, multi-ring (MtRg), hybrid, tree, and star topology in order.

In simulations, the capacities of 8 edge servers are set as $4, 4, 4, 4, 4, 6, 6, 8$ and the weight parameters are set as $\alpha = \beta = \gamma = \zeta = \delta = 1$. We perform simulations on the number of chains m from 1 to 5 with 5 VNFs for each chain. We adopt the Mixed-Integer Programming (MIP) solver to find OPT for SFCD. Note that the practical approximation ratios of ECNF or OCNF in Fig. 3 are computed by dividing the total cost of ECNF or OCNF by that of OPT, computed by MIP solver.

2) Approximation Ratio Verification of ECNF: As we can see from Fig. 3(a), the median ratios of ECNF under different network topologies are all near 1.08 and the average ratios are around 1.13, significantly smaller than the upper bound proved in Theorem 5. In Fig. 3(a), we find different topologies do not make a big difference in the performance of ECNF, which indicates ECNF can adapt to various network topologies.

Fig. 3(b) exhibits the practical ratios of ECNF when deploying different numbers of SFCs. The results for 1 and 2 SFCs are the edge-only cases when all SFCs are successfully deployed at the edge and ECNF is reduced to CNF. The results for 3-5 SFCs reveal the critical cases when ECNF may need cloud resources while OPT does not. And the results for deploying 6-9 SFCs demonstrate the cases when there are no sufficient edge resources and cloud resources must be used even for OPT. As shown in Fig. 3(b) the worst case of approximation ratios of ECNF is achieved under the critical cases when the edge resources are just sufficient for OPT but insufficient in terms of ECNF. It is consistent with the expression of r in Theorem 5.

3) Approximation Ratio Verification of OCNF: As shown in Fig. 3(c), the median approximation ratios of OCNF under various network topologies are all near 1.15, while the average ratios around 1.25, also far smaller than the upper bound given in Theorem 9. We can see different topologies do not make a big difference in the performance of OCNF, meaning OCNF can fit various network topologies.

Fig. 3(d) displays the performance of OCNF when deploying different numbers of SFCs. Similar to ECNF, the results for 1 and 2 SFCs demonstrate the edge-only cases; the results for 3-5 SFCs reveal the critical cases when OCNF may need cloud resources while OPT need not; while the results for 6-9 SFCs are the cases there are no sufficient edge resources

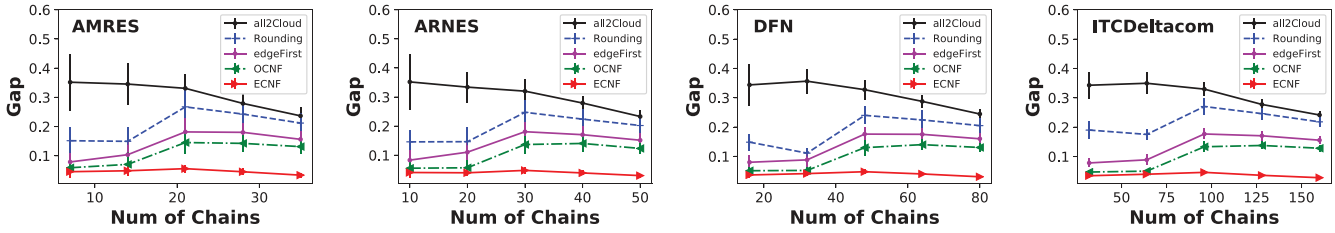
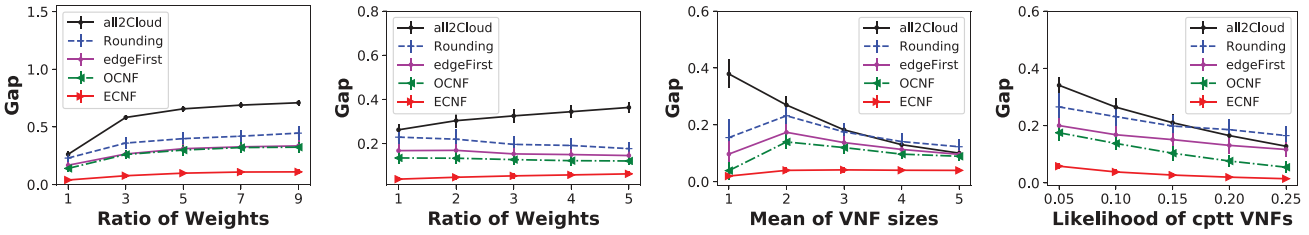


Fig. 5. Performance comparisons of different algorithms on 4 different real network topologies.



(a) Ratio $= \frac{\beta}{\alpha} = \frac{\zeta}{\alpha}$ (b) Ratio $= \frac{\gamma}{\alpha}$ (c) Mean values of VNF sizes (d) likelihood of computation-intensive VNFs with $t_{i,j} > 1$ ms

Fig. 6. Performance comparisons of different algorithms with different settings.

and cloud resources must be used even for OPT. The worst performance of OCNF is achieved under the critical cases when the edge is just just sufficient for OPT but insufficient in terms of OCNF, which is consistent with the expression of r in Theorem 9

C. Simulations on 4 Large-Scale Real Network Topo.s

1) Network Topology and Server Capacity Setup: Moreover, we also conduct simulations on 4 larger real network topologies from the Internet topology zoo [31]: (1) AMRES (25 nodes and 24 links), (2) ARNES (34 nodes and 46 links), (3) DFN (58 nodes and 87 links), (4) ITCDeltacom (113 nodes and 160 links).

In simulations, we set the capacities of edge servers following the normal distribution of a mean of 12 and a standard variance of 4, with the limitation of no less than 6. And each SFC has 10 VNFs. If not mentioned, we default the weight parameters as $\alpha = 1, \beta = 10, \gamma = 2, \zeta = 10$. (Note that below we will also test the different parameter settings in Fig. 6.)

In this part, we will compare our proposed ECNF and OCNF with the following three benchmarks: (1) **Rounding Algorithm**: a classical approach to obtain a provable bound for ILP. See [9], [16], [32] as examples; (2) **edge-first strategy**: an intuition edge-based SFC deployment employed by work [4], [5], where SFCs are offloaded to the cloud when the deployment exceeds the capabilities of the edges; (3) **all-to-cloud strategy**: a baseline that offloads all SFCs to the cloud.

Besides, in each trial of simulations, we also compute the lower bound of SFCD by *Gubori_objbound* so that we can compute the gaps of these algorithms, i.e., $\frac{Algo\_ObjVal - objbound}{Algo\_ObjVal}$, which clearly show the performance of each algorithm. In all plots below, we adopt the average value from 100 groups of simulations to show average cases. And the errors shown on plots are determined by the standard variances of the 100 groups of simulations with the same setting.

TABLE II
TIME COST OF DIFF. ALGORITHMS ON ITCDELTACOM

m	20	40	60	80	100
Rd	37.10	78.70	116.8	164.4	246.2
ECNF	0.01639	0.01900	0.02598	0.04716	0.04902
OCNF	0.01079	0.01164	0.01206	0.01262	0.01324

2) Performance Comparisons With Benchmarks: In what follows, we show simulation results on 4 different real network topologies with different m. As shown in Fig. 5, ECNF and OCNF always perform better than the benchmarks, which indicates the superiority of our two designed algorithms.

3) Time Cost Comparisons With Benchmarks: In Table II, we list the time cost of different algorithms running on the ITCDeltacom topology with 113 nodes. From these data, we can get the conclusion that ECNF and OCNF are both very fast algorithms. They have a dramatic advantage on the time cost compared with the popularly-used rounding algorithm.

4) Performance Comparisons With Different Settings: Next, we assess the robustness of our proposed algorithms under varied parameter configurations and present the results in Fig. 6. The simulations involve 25 SFCs, the AMRES network topology, and diverse parameter settings. It's worth noting that parameters maintain their original configurations unless explicitly stated otherwise. First, we examine the influence of objective weights by adjusting relative weights between resource cost and communication latency, depicted in plot (a), as well as between edge and cloud resources, illustrated in plot (b). Additionally, we explore the impact of VNF parameters by adjusting mean values of VNF sizes, shown in plot (c), and the appearance likelihood of computation-intensive VNFs

with $t_{i,j} > 1$ ms, depicted in plot (d). Fig. 6 shows that different parameter settings do not make a big difference in the performance of the algorithms, and the proposed ECNF and OCNF algorithms always outperform the benchmarks under all the settings.

VIII. Conclusion

In this paper, we investigate the SFCD problem in the edge-cloud continuum, with the goal of jointly minimizing resource consumption and network latency. We propose two approximation algorithms, OCNF and ECNF, for online and offline versions of the SFCD problem, respectively. The proposed algorithms aim to balance resource and latency while optimizing the sweet-spot of resource cost on both the edge and the cloud, as well as all corresponding network latency. Specifically, the Next Fit (NF) strategy employed in these two algorithms ensures efficient resource utilization and avoids redundant data traffic. Also, the designed corresponding sub-algorithms for virtual network embedding, i.e., DST for CNF, the revised DST for ECNF, and HPT for OCNF, play a critical role in reducing latency. In addition, we prove the proposed ECNF and OCNF algorithms have constant approximation ratios. Simulation results verify the proved theoretical bounds and using real-world network topologies show the proposed algorithms outperform popular baselines.

In future discussions, we will try to refine the online algorithm to enhance its real-time decision-making capabilities in response to the dynamic nature of network environments. We also plan to delve into extreme scenarios, such as cases involving bandwidth overload, and explore additional optimization goals for a more comprehensive analysis.

References

[1] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Piscataway, NJ, USA: IEEE Press, 2014, pp. 7–13.

[2] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 693–701.

[3] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Oct. 2019.

[4] J. Son and R. Buyya, "Latency-aware virtualized network function provisioning for distributed edge clouds," *J. Syst. Softw.*, vol. 152, pp. 24–31, Jun. 2019.

[5] J. Martín-Pérez, F. Malandrino, C.-F. Chiasserini, and C. J. Bernardos, "OKpi: All-KPI network slicing through efficient resource allocation," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 804–813.

[6] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 267–276.

[7] H. Ren et al., "Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2050–2066, Sep. 2020.

[8] Y. Mao, X. Shang, and Y. Yang, "Near-optimal resource allocation and virtual network function placement at network edges," in *Proc. IEEE 27th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 18–25.

[9] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 445–459, Feb. 2021.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[11] Y. Liu, Y. Mao, X. Shang, Z. Liu, and Y. Yang, "Distributed cooperative caching in unreliable edge environments," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 1049–1058.

[12] Z. Wan, "Cloud computing infrastructure for latency sensitive applications," in *Proc. IEEE 12th Int. Conf. Commun. Technol.*, Piscataway, NJ, USA: IEEE Press, 2010, pp. 1399–1402.

[13] Y. Mao, X. Shang, and Y. Yang, "Provably efficient algorithms for traffic-sensitive SFC placement and flow routing," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 950–959.

[14] D. S. Johnson, "Near-optimal bin packing algorithms," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1973.

[15] M. A. Khoshkholghi et al., "Service function chain placement for joint cost and latency optimization," *Mobile Netw. Appl.*, vol. 25, pp. 2191–2205, Dec. 2020.

[16] X. Shang, Z. Liu, and Y. Yang, "Network congestion-aware online service function chain placement and load balancing," in *Proc. 48th Int. Conf. Parallel Process.*, 2019, pp. 1–10.

[17] D. Li et al., "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[18] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2017, pp. 1–9.

[19] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CU allocation in 5G," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 1943–1951.

[20] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2994–3008, Aug. 2022.

[21] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao, "Towards latency optimization in hybrid service function chain composition and embedding," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1539–1548.

[22] V. Valls, G. Iosifidis, G. De Mel, and L. Tassiulas, "Online network flow optimization for multi-grade service chains," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1329–1338.

[23] N. He et al., "Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1186–1201, Apr. 2023.

[24] "Network functions virtualization—Introductory white paper," ETSI, SDN and OpenFlow World Congress, Darmstadt, Germany, Oct. 2012, pp. 1–16. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[25] Y. Mao, X. Shang, and Y. Yang, "Joint resource management and flow scheduling for SFC deployment in hybrid edge-and-cloud network," in *Proc. IEEE INFOCOM—Conf. Comput. Commun.*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 170–179.

[26] H. Fleischner, "The square of every two-connected graph is Hamiltonian," *J. Combinatorial Theory, Ser. B*, vol. 16, no. 1, pp. 29–34, 1974.

[27] H. T. Lau, "Finding a Hamiltonian cycle in the square of a block," Ph.D. dissertation, McGill Univ., Montreal, QC, Canada, 1980.

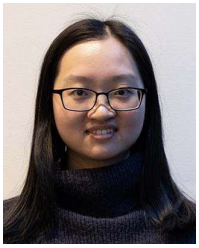
[28] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Softw.: Pract. Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[29] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A blockchain-based lightweight framework for Edge and Fog computing," *J. Syst. Softw.*, vol. 154, pp. 22–36, Aug. 2019.

[30] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ, USA: Wiley, 2005.

[31] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

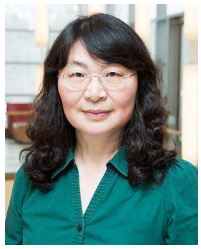
[32] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 1346–1354.



Yingling Mao (Graduate Student Member, IEEE) received the B.S. degree in mathematics and applied mathematics from Zhiyuan College, Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, edge computing, cloud computing, and quantum networks.

Xiaojun Shang received the B.Eng. degree in information science and electronic engineering from Zhejiang University, Hangzhou, China, and the M.S. degree in electronic engineering from Columbia University, New York, NY, USA, and the Ph.D. degree in computer engineering from Stony Brook University. He is currently an assistant professor with the Department of Computer Science and Engineering at the University of Texas, Arlington. He is currently working toward the Ph.D. degree in computer engineering with Stony Brook University. His research interests include cloud computing and data center networks, with focus on placement and routing of virtual network functions and resilience of service function chains.

Yu Liu (Graduate Student Member, IEEE) received the B.Eng. degree with honor in telecommunication engineering from Xidian University, Xi'an, China. He is currently working toward the Ph.D. degree in computer engineering with Stony Brook University. His research interests include online algorithms, distributed storage, cloud computing, edge computing, and data center networks, with focus on placement and resource management of virtual network functions and reliability of service function chains.



Yuanyuan Yang (Life Fellow, IEEE) received the B.Eng. and M.S. degrees in computer science and engineering from Tsinghua University, and the M.S.E. and Ph.D. degrees in computer science from Johns Hopkins University. She is currently a Distinguished Professor with Stony Brook University. Her research interests include edge computing, data center networks, cloud computing, and wireless networks. She has published over 480 papers and holds seven U.S. patents in these areas. She is the Editor-in-Chief for IEEE TRANSACTIONS ON CLOUD COMPUTING and an Associate Editor for *ACM Computing Surveys*. She has served as the Associate Editor-in-Chief for IEEE TRANSACTIONS ON COMPUTERS and an Associate Editor for IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. She has also served as a Program Director with the National Science Foundation and as the General Chair, Program Chair, or Vice Chair for several major conferences.