Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge

Panpan Jin[1] Xincai Fei[1] Qixia Zhang[1] Fangming Liu*[1] Bo Li[2]
[1]National Engineering Research Center for Big Data Technology and System,
Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]Hong Kong University of Science and Technology, Hong Kong

***Abstract*—With the increasing demand of low-latency network services, mobile edge computing (MEC) emerges as a new paradigm, which provides server resources and processing capacities in close proximity to end users. Based on network function virtualization (NFV), network services can be flexibly provisioned as virtual network function (VNF) chains deployed at edge servers. However, due to the resource shortage at the network edge, how to efficiently deploy VNF chains with latency guarantees and resource efficiency remains as a challenging problem. In this work, we focus on jointly optimizing the resource utilization of both edge servers and physical links under the latency limitations. Specifically, we formulate the VNF chain deployment problem as a mixed integer linear programming (MILP) to minimize the total resource consumption. We design a novel two-stage latency-aware VNF deployment scheme: highlighted by a constrained depth-first search algorithm (CDFSA) for selecting paths, and a path-based greedy algorithm (PGA) for assigning VNFs by reusing as many VNFs as possible. We demonstrate that our proposed algorithm can efficiently achieve a near-optimal solution with a theoretically-proved worst-case performance bound. Extensive simulation results show that the proposed algorithm outperforms three previous heuristic algorithms.**

I. Introduction

Recent years, with the increasing popularity of mobile devices, service demands from mobile users grow exponentially. The emerging service requirements from applications such as virtual reality [1], Internet of Things [2] and wearable devices, are forming the majority of computation in the network [3]. Moreover, as the development of these applications, users have a strong desire for high-speed, low-latency (tens of milliseconds or less), always-on and multimedia-oriented connections [4], [5].

To accommodate this trend, telecommunication service providers introduce mobile edge computing (MEC), a new technology that provides an IT service environment and cloud-computing capabilities at the edge of the network, within close proximity to mobile users [6]. With this vision, the computation tasks can be offloaded from the core network to the edge, which is highly beneficial to reducing the end-to-end latency, ensuring service delivery, offering an improved user experience, and cutting down the resource utilization in the core network [6]. Meanwhile, a variety of devices such as base stations [7], enterprise gateways and home routers [8] are increasingly equipped with computation capabilities at the network edge, which makes MEC a hot research topic in both academia and industry [9], [10], [11], [12].

To provide flexible services with both reduced capital and operational expenses, another promising technology called network function virtualization (NFV) [13] has been widely advocated by service providers. In the context of MEC, network services can be flexibly provisioned using virtual network functions (VNFs) both in the back-end cloud (core network) and at the network edge, while serving the users in an on-demand fashion. Besides, service demands at the edge usually require rigorous low-latency in order to offer an instantaneous experience to mobile users. For instance, service demands from augmented reality require that the end-to-end latency should not exceed 10-20 ms [14]. As a typical application in NFV, a network service chain generally consists of an ordered sequence of VNFs. In practice, VNFs cannot only be shared by different service demands, but also can be deployed on different network devices. As a result, both computing and bandwidth resources can be efficiently utilized. Hence, in this work, we focus on an important problem: *How to efficiently utilize the computing resources of edge devices while reducing bandwidth consumption of links, by taking the reuse of VNF instances and latency requirements into account?*

However, it is very challenging to simultaneously achieve the above joint objectives since there exists a dilemma between the resource utilization of the servers and links. On one hand, in order to reserve as many bandwidth resources as possible, the service chain should be routed along the shortest path. As a result, some VNF instances located on other paths are too costly to be reused, thus new instances have to be launched along the shortest path to support the demands. On the other hand, if we focus on reusing as many launched VNFs as possible, a longer path might be selected for the service chain with more bandwidth resources consumed, and a higher chance to violate the latency limitations. For example, a service chain

*The corresponding author is Fangming Liu (fmliu@hust.edu.cn). This work was supported in part by the NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005, in part by National Key Research & Development (R&D) Plan under grant 2017YFB1001703, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals, and in part by RGC GRF grants under the contracts 16206417 and 16207818, Guangzhou Development Zone Project Grant 2017GH23.

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

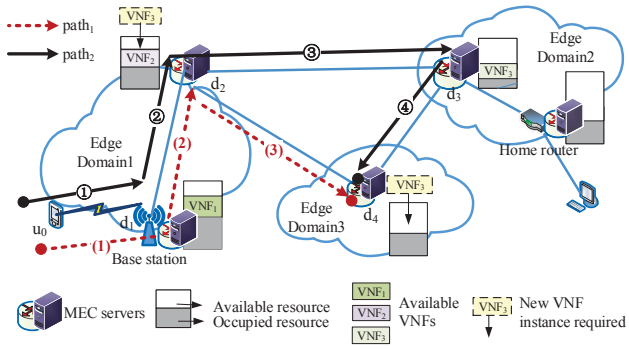


Fig. 1. The architecture of deploying VNF chains at the network edge

with a flow rate r from server d_1 to d_4 requires to traverse VNF_1, VNF_2, VNF_3 in sequence. In case 1) when we choose the shortest path_1 shown as the red dashed line in Fig. 1, then the bandwidth consumption $3 * r$ is the minimum. Since path_1 does not contain VNF_3, we need to launch a new VNF_3 instance on d_2 or d_4, which will consume extra server resources. In case 2) if we try to reuse all reusable launched VNF instances, we choose path_2, the black solid line in Fig. 1, which contains all the required VNFs. In this case, there will be no extra server resource consumption, but at the cost of consuming more bandwidth resource $(4 * r)$ and increasing the end-to-end latency. Therefore, it is very difficult to reach an optimal deployment scheme for service chains while jointly considering the two cases above.

To tackle the above challenges, in this paper, we formulate the VNF chain deployment problem at the network edge as an optimization problem, aiming to minimize the resource consumption of both servers and links with latency guarantees. In the scenario of the network edge, service demands from multiple users might simultaneously arrive at the access points (APs) waiting to be served. Thus, we consider the queueing delay in our model, which is different from many schemes in data centers [15]–[17]. Since the problem is proved to be NP-hard, we seek to find a deployment scheme for service chains in sequence and divide the original problem into two subproblems, i.e., selecting paths first and then assigning VNFs. After that, we devise a two-phase near-optimal solution: 1) a constrained depth-first search algorithm (CDFSA) for deriving the set of feasible paths, and 2) a path-based greedy algorithm (PGA) for assigning VNFs by reusing VNFs as many as possible. We also derive the worst-case performance bound of our proposed algorithms. Finally, extensive simulation results demonstrate that the proposed algorithms can efficiently solve the VNF chain deployment problem, and outperform three heuristic algorithms. The main contributions are as follows:

- Towards the network edge, we present an optimization formulation for the VNF chain deployment problem that minimizes the resource consumption of both servers and links with the latency guarantees. Particularly, we take the queuing delay at the APs into account for meeting the latency requirements.
- We seek to find a deployment scheme for service chains in sequence and divide the original problem into two subproblems, i.e., selecting paths first and then assigning VNFs, and propose a two-phase near-optimal algorithm to efficiently solve the problem.
- We provide theoretical analysis for the proposed algorithm with a worst-case performance bound. Extensive simulations based on real-world topologies demonstrate that our proposed algorithm can efficiently and effectively obtain a solution, and outperform three existing heuristic algorithms in terms of resource consumption.

II. Related Work

With the increasing popularity of NFV, the traditional service provision problem within the telecommunications industry has been migrated to the NFV scenario. In short, there is a continuous effort to investigate the problem from different angles [4], [18]–[23]. Among them, [18]–[21] tackle the VNF placement and routing problem, which is a significant problem arisen from NFV.

A. VNF Placement and Routing in General

Most existing works focus on the VNF placement and routing problem with different objectives and propose corresponding solutions. For example, Fei et al. [24] investigate the problem for VNF providers to dynamically scale the VNF deployments and reroute traffic demands. They first employ an online learning approach to predict the arriving traffic flows and then propose a joint online algorithm for new VNF instance assignment and service chain rerouting. Sang et al. [20] tackle the VNF placement problem to minimize the total number of VNF instances in a network reduced from the set cover problem. They design two greedy algorithms to solve the problem, both of which are proven to be asymptotically optimal.

There are also some works targeting on simultaneously optimizing the consumption of both the computing and bandwidth resources. Cohen et al. [21] address the VNF placement problem within physical networks, which aims at minimizing the total setup costs of functions and the distance costs between the clients and the servers. They provide a near-optimal approximation algorithm with theoretically proven performance. However, they assume that each demand of service only requires a single VNF rather than a service chain. In [18], the authors discover the resources relation between links and servers, and study the joint problem of VNF placement and path selection to better utilize both the network and server resources, whose objective is to maximize the total size of admitted demands rather than optimizing resource utilization, and they ignore the service latency requirements.

B. VNF Placement and Routing at the Network Edge

The above works all focus on a cloud scenario, which cannot be directly applied to the network edge, because of the characteristics of the network edge, such as resource shortage and the multiple access points for users. There also exist a few works tackling the VNF placement (and routing) problem at the network edge so far. For example, [25] studies the

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

TABLE I
KEY NOTATIONS.

Notation	Definition
$\mathbb{G} = (\mathbb{U}, \mathbb{D}, \mathbb{E})$	The graph of the network
$\mathbb{S}$	The set of VNF service chains, $S_i \in \mathbb{S}$ is the service chain from user $u_i \in \mathbb{U}$
$e(u,v)$	The available bandwidth resource on link (u,v)
$B_{(u,v)}$	The bandwidth capacity of link $e(u,v) \in \mathbb{E}$
$l_{i(u,v)}$	The latency of the link $e(u,v)$ when S_i using it.
C_j	The resource capacity of hardware device $d_j \in \mathbb{D}$
R_n	The resource demand of a type-n VNF instance
W_n	The processing capacity of a type-n VNF instance
β_i	The maximum latency constraint of $S_i \in \mathbb{S}$
(ϕ_i, t_i, r_i)	The three-tuple of service chain S_i; $\phi_i \subset \mathbb{D}$ is the set of available access points of S_i; $t_i \in d_j$ denotes the destination of S_i, and r_i is the flow rate of S_i
$X_{i,j}^{l}$	Whether the l^{th} VNF of service chain S_i is served by server d_j
$Z_{i(u,v)}^{ll'}$	Whether the subchain between the l^{th} and l'^{th} VNF of S_i is routed on link $e(u,v)$
$T_{i,n}^{l}$	Whether the l^{th} VNF of the service chain $S_i \in \mathbb{S}$ requires the type-n VNF
$Y_{j,n}$	The number of type-n VNF instances in server d_j

problem of jointly optimizing the access network selection and service placement problem for MEC. They design an online framework to improve the QoS of the services, but each service only contains a single VNF. Zhang et al. [26] focus on the VNF placement problem in 5G service-customized network slices, they propose a general 5G network slice framework, taking the VNF interference into account, and propose an adaptive interference-aware heuristic (AIA) approach to automatically place VNFs in each 5G network slice. However, they do not consider the resource optimization at all. Cziva et al. [27] follow the work [4] to investigate the latency optimization of single VNF placement and routing problem without considering the reusability of the VNF instances. In addition, they emphasize the limited resource provision at the network edge but do not take the resource optimization into consideration.

To the best of our knowledge, the efficient VNF chain deployment problem for resource optimization with latency limitations at the network edge has not been investigated before. Hence, in this paper, our objective is to simultaneously reuse the server resources and reduce the bandwidth consumption while providing latency guarantees.

III. MODEL AND FORMULATION

A. The Network Model

We model a network as an undirected graph $\mathbb{G} = (\mathbb{U}, \mathbb{D}, \mathbb{E})$, where $\mathbb{U}, \mathbb{D}$ and $\mathbb{E}$ denote the set of users, the set of hardware servers (including all kinds of devices equipped with computation capabilities in the network edge, such as the access points, the switches, the routers as shown in Fig. 1), and the set of physical links between the servers, respectively. We assume each server $d_j \in \mathbb{D}$ has a resource capacity denoted by C_j, i.e., the amount of available resources in terms of CPU and memory. Similarly, each link $e(u,v) \in \mathbb{E}$ between server $d_u \in \mathbb{D}$ and server $d_v \in \mathbb{D}$ has a bandwidth capacity denoted by $B(u,v)$.

Say there are different service demands from different users, and each of them requires a service chain consisting of an ordered sequence of VNFs. We use $\mathbb{N}$ and $\mathbb{S}$ to denote the set of different types of VNFs and the set of different service chains, respectively. For each user $u_i \in \mathbb{U}$, we denote $S_i \in \mathbb{S}$ as the requested service chain, and $|S_i|$ is the length (the number of the VNFs) of S_i. We next assume that $s_i^l \in S_i$ is the l^{th} VNF of service chain S_i, and $\Phi_i^{ll'}$ is the subchain between the $S_i{}^l$ and $S_i{}^{l'}$ $(1 \le l < l' \le |S_i|$, $\Phi_i^{ll'} \subseteq S_i)$. Specifically, we introduce $T_{i,n}^{l}$ to indicate whether the l^{th} VNF of the service chain S_i is type-$n \in \mathbb{N}$. In reality, a user might require more than one service chain, which can be regarded as different users in the same physical location require different service chains.

We further define a three-tuple (ϕ_i, t_i, r_i) for service chain S_i to indicate the source, the destination, and the flow rate of S_i, respectively. Specifically, ϕ_i is the set of the nearby APs collocated to different edge clouds, which are in close proximity to u_i, which means S_i can access the network from the k^{th} AP $d_{i,k} \in \phi_i$. Since each $d_{i,k} \in \phi_i$ corresponds to a server $d_j \in \mathbb{D}$, such as the base station as shown in Fig. 1, we simply denote by α_j the processing capacity of AP $d_j \in \mathbb{D}$. Here, we assume that the flow rate of a service chain always remains unchanged when traversing different servers, since traffic change effects have been investigated in [28], and our formulation can easily extend to the changing case.

Lastly, let $\mathbb{P}$ be the set of the physical paths on which service chains are routed along. Each $P_i \in \mathbb{P}$ consists of an ordered sequence of physical links for service chain S_i. For instance, $\text{Path}_2 = \{e(d_1,d_2), e(d_2,d_3), e(d_3,d_4)\}$ in Fig. 1 indicates that the routing path of the service chain S_i contains four server nodes d_1, d_2, d_3 and d_4.

B. Problem Formulation

We assume all the VNFs can be shared by different service chains so that their residual processing capacities can be maximally reused. We denote by $Y_{j,n}$ the number of type-$n \in \mathbb{N}$ VNF instance in server $d_j \in \mathbb{D}$. Each time when a service chain requires a type-n VNF from d_j, if there is reusable type-n VNF instance, $Y_{j,n}$ will remain unchanged; else we will launch a new type-n instance in d_j, and $Y_{j,n}$ becomes $Y_{j,n} + 1$. When launching a type-n VNF instance, it requires R_n device resource. To guarantee that the total resource consumption of each server d_j should not exceed the resource capacity C_j, we have:

$$\sum_{n \in \mathbb{N}} Y_{j,n} R_n \le C_j, \quad \forall d_j \in \mathbb{D}. \tag{1}$$

We use $Z_{i(u,v)}^{ll'}$ to represent whether the subchain $\Phi_i^{ll'}$ of S_i traverses link $e(u,v)$. In the same way, the sum of throughput of all the traffic flows along link $e_{(u,v)} \in \mathbb{E}$ should not exceed

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

its bandwidth capacity $B(u,v)$, which can be represented as follows:

$$\sum_{S_i \in \mathbb{S}} \sum_{\Phi_i^{ll'} \subseteq S_i} Z_{i(u,v)}^{ll'} \cdot r_i \leq B(u,v), \quad \forall e(u,v) \in \mathbb{E}, 1 \leq l < l' \leq |S_i|. \tag{2}$$

Let W_n be the processing capacity of each type-n VNF instance. To ensure that all the VNF instances on any server d_j have sufficient capacities to process the flows, we have:

$$\sum_{S_i \in \mathbb{S}} \sum_{s_i^l \in S_i} T_{i,n}^l \cdot X_{i,j}^l \cdot r_i \leq Y_{j,n} \cdot W_n, \forall n \in \mathbb{N}, \forall d_j \in \mathbb{D}, \tag{3}$$

where $X_{i,j}^l$ denotes whether the l^{th} VNF of service chain S_i is served by server d_j.

As mentioned above, we assume that each service chain S_i carries a single flow, i.e., each $s_i^l \in S_i$ just gets served by one server. Then, we have the following constraint for $X_{i,j}^l$:

$$\sum_{d_j \in \mathbb{D}} X_{i,j}^l = 1, \quad \forall S_i \in \mathbb{S}, \forall l \in [1, |S_i|]. \tag{4}$$

Next, for each service chain S_i, if it can be successfully accepted, we should make sure the total latency will not exceed its latency limitation β_i. Here, we consider two kinds of latency, namely the queueing delay in the AP $d_j \in \phi_i$ selected by S_i, and the propagation delay and transmission delay $l_{i(u,v)}$ on the link $e_{(u,v)}$. To analyze the queueing delay of users at the AP, we model each AP as an M/M/1 queue. By applying the Little's law, the average queueing delay in d_j can be calculated as follows:

$$q_j = \frac{1}{\alpha_j - \sum_{S_i \in \mathbb{S}} A_{i,j} \cdot r_i}, \tag{5}$$

where $A_{i,j}$ denotes whether the service chain S_i chooses the AP $d_j \in \phi_i$ to access the edge network. Then we have:

$$\sum_{d_j \in \mathbb{D}} A_{i,j} \cdot q_j + \sum_{\Phi_i^{ll'} \subseteq S_i} Z_{i(u,v)}^{ll'} \cdot l_{i(u,v)} \leq \beta_i, \quad \forall e(u,v) \in \mathbb{E}, \forall S_i \in \mathbb{S},\ 1 \leq l < l' \leq |S_i|, j = u \text{ if } l = 1. \tag{6}$$

Since S_i can only choose one AP each time, we have the following constraint for $A_{i,j}$:

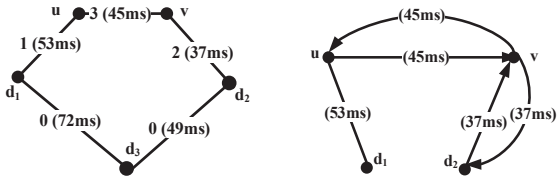
$$\sum_{d_j \in \mathbb{D}} A_{i,j} = 1, \quad \forall S_i \in \mathbb{S}. \tag{7}$$

Last but not least, we should guarantee all the VNFs in S_i will be served, which can be represented as follows:

$$\sum_{s_i^l \in S_i} \sum_{d_j \in \mathbb{D}} X_{i,j}^l = |S_i| \quad \forall S_i \in \mathbb{S}. \tag{8}$$

Our objective is to jointly minimize the resource consumption on servers and links. Since the device resource consumption at server d_j is $\sum_{n \in \mathbb{N}} Y_{j,n} R_n$, and the total bandwidth consumption of the link $e_{(u,v)}$ is $\sum_{S_i \in \mathbb{S}} \sum_{\Phi_i^{ll'} \subseteq S_i} Z_{i(u,v)}^{ll'} \cdot r_i$. Then the optimization problem of the joint resource consumption can be formulated as follows:

$$\min \sum_{d_j \in \mathbb{D}} \sum_{n \in \mathbb{N}} Y_{j,n} \cdot R_n + \sum_{e(u,v) \in \mathbb{E}} \sum_{S_i \in \mathbb{S}} \sum_{\Phi_i^{ll'} \subseteq S_i} Z_{i(u,v)}^{ll'} \cdot r_i \quad \text{s.t.} \quad (1), (2), (3), (4), (5), (6), (7), (8) \tag{9}$$



(a) The undirected graph $\mathbb{G}'$ (b) The reconstructed graph $\mathbb{H}$

Fig. 2. Reconstruct a new graph $\mathbb{H}$ according to the repeating times of the links. In (a), the numbers outside parentheses denote the repeating times, the numbers inside parentheses denote the latency of the links. The numbers in (b) denote the latency of the links.

IV. Two-phase Near-optimal Solution

A. Problem Complexity

The VNF deployment problem has been proved to be NP-hard in many previous works [17], [29]–[32]. Here, we relax the formulation of our VNF chain deployment problem exhibited in Eq. (9) with only server resource constraints defined in Eq. (1), Eq. (3) and Eq. (4), and VNF constraint in Eq. (8), which then becomes a reduction form of a classical bin packing problem. Many existing works have proved the NP-hardness of the bin packing problem, such as [33]–[35]. Now that with additional constraints in Eq. (2), Eq. (5), Eq. (6), Eq. (7), we can conclude that the VNF chain deployment problem in Eq. (9) is also NP-hard.

B. An Overview of Solution

Since the VNF deployment problem is proved to be NP-hard, it is computational expensive to derive the optimal solution for deploying all service chains. By making an assumption that a sequence of service chains arrive into the system one by one without the knowledge of future arrivals, our problem focuses on how to deploy and route a single service chain to minimize the resource consumption.

However, it is still tricky to solve the problem. Thus, we seek to divide the original single chain problem into two subproblems, i.e., selecting paths first and then assigning VNFs. Specifically, in the first subproblem, for service chain $S_i = (\phi_i, t_i, r_i)$, we find all the feasible paths from source ϕ_i to destination t_i which are called pre-routing paths, under the constraints of the bandwidth of links and the latency of the service chain. We then put the paths into a set of paths denoted by P_i^{Λ}. After that, to solve the second subproblem, we take turns to pick the paths out of P_i^{Λ}. For each $p \in P_i^{\Lambda}$, considering the reusable VNF instances and the residual resource capacities of the servers, we assign VNFs along the path with the maximum reusing number. Next, we choose an assignment scheme with minimum resource consumption among all the paths. This ensures our solution obtained by the two-phase strategy for single chain is near-optimal. In the end, we add up all the resource consumption of single chains to get the total resource consumption of all the arriving service chains.

C. Selecting Paths

For the first subproblem, we construct a new weighted undirected graph $\mathbb{G}' = (\mathbb{D}, \mathbb{E}; w_1, w_2)$ (see Fig. 2(a)), where

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

w_1 and w_2 denote the delay and residual bandwidth of each link, respectively. In other words, given a graph $\mathbb{G}'$ and a certain service chain S_i with latency limitations, our objective is to find all the feasible paths in $\mathbb{G}'$, which consist of a sequential set of links from ϕ_i to t_i without exceeding the bandwidth capacity limitations.

Reconstruct a new network graph. To describe the available servers and links for service chain $S_i = \{s_1, s_2, ..., s_{|S_i|}\}$, we construct a new weighted mixed multi-graph $\mathbb{H} = (\mathbb{D}', \mathbb{E}'; w)$ (see Fig. 2(b)) from graph $\mathbb{G}'$, where the weighted value of link $e(u, v)$ is $l_{i(u,v)}$. Hereinafter, we detail the construction of graph $\mathbb{H}$ in three steps:

Step 1: **Servers deriving.** Firstly, we add all the available APs $d_j \in \phi_i$ of S_i to $\mathbb{H}$. We then assume that the minimum resource consumption among all the VNF $s_l \in S_i (l \in \{1, ..., |S_i|\})$ instance is R^i_{min}. If the residual resource on server $d_j \in \mathbb{D}$ is more than R^i_{min}, or there are any available VNF instances could be reused by VNF $s_l \in S_i (l \in \{1, ..., |S_i|\})$, then we add server d_j to $\mathbb{H}$.

Step 2: **Links deriving.** There are chances that a VNF of a chain will be served by a server where the upstream VNF has been served, as long as the server has sufficient resources left, which means there might exist loops in the routing paths. Meanwhile, if the bandwidth capacity is sufficient, each link of the graph could be traversed multiple times. For example, given $S_i = (d_0, \gamma, d_4) = \{s_0, s_1, s_2, s_3\}$ and a part of network $\mathbb{G}'$ (see Fig. 3(a)), if we deploy s_0, s_1, s_2 and s_3 on server d_1, d_2, d_3 and d_2, respectively, the traffic flow will pass through the link (d_2, d_3) twice, considering the links in $\mathbb{G}'$ are undirected.

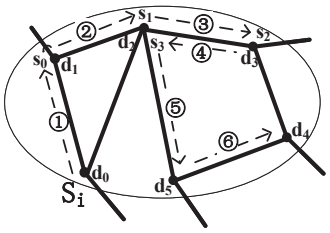
Since our objective is to minimize resource consumption with latency constraints at the network edge, it is unreasonable for a service chain to pass through a link time and again. Thus, we assume a service chain traverses the same link at most twice (in different directions). For example, as shown in Fig. 2, for service chain S_i, if the permitted maximum repeating times of the link between node u and v is $\lfloor \frac{B_(u,v)}{r_i} \rfloor > 1$ (where $B(u, v)$ is the residual bandwidth capacities between node u and v), then two directed links $e(u, v)$ and $e(v, u)$ with a weighted value as $l_{i(u,v)}$ will be added to graph $\mathbb{H}$. If $\lfloor \frac{B_(u,v)}{r_i} \rfloor = 1$, then an undirected link $e(u, v)$ with a weighted value as $l_{i(u,v)}$ will be added to $\mathbb{H}$.

Step 3: **Dissociative servers pruning.** Up to now, if there are any servers in $\mathbb{H}$ without links connected to, just remove them.

A Constrained Depth First Search Algorithm (CDFSA). With the given graph $\mathbb{H}$, we will continue to use CDFSA to compute the set of pre-routing paths P^Λ_i for service chain S_i, as exhibited in Algorithm 1.

Firstly, we employ an adjacency matrix M to store both the directed and undirected links in $\mathbb{H}$. Specifically, we use 0, 1 and 2 to indicate no links, undirected links and directed links, respectively. For a pair of nodes u and v, we have:

$$M[u][v] = M[v][u] = \begin{cases} 0, \text{ if there is no link,} \\ 1, \text{ if there is an undirected link,} \\ 2, \text{ if there are two directed links.} \end{cases} \quad (10)$$



Server	Reusable VNF(s)
d_0	s_3
d_1	s_0,s_2,s_3
d_2	s_1,s_3
d_3	s_0,s_2
d_4	s_1
d_5	s_0,s_2

(a) The deployment of service chain S_i (b) The reusable VNFs

Fig. 3. The solid lines and the dotted lines in (a) denote the physical links and the flow of service chain S_i respectively. The data in (b) indicates the reusable VNF instances for S_i at each server, which is derived from Eq. (11).

Algorithm 1: A Constrained Depth First Search Algorithm (CDFSA)

Input: $\mathbb{H}$, (ϕ_i, r_i, t_i)
Output: P^Λ_i

Initialize the adjacency matrix M and weighted value matrix E,
initialize a stack called $c_p = \varnothing$ to store the nodes of the path.
for *each* $d_j \in \phi_i$ **do**
  $u \leftarrow d_j$;
  Calculating the queuing delay q_j of d_j as Eq. (5)
  $L_t \leftarrow q_j$;
  function Selection_Paths(u, t_i, L_t, β_i)
    push the current node u into stack c_p.
    if *the current node* u == *end point* t_i **then**
      Fetch the path from c_p and put it into the set P^Λ_i;
      return;
    end
    for *each node* v *in graph* H **do**
      if $M[u][v] > 0$ && $L_t + E[u][v] < \beta_i$ **then**
        if $M[u][v] == 2$ **then**
          $M[u][v] \leftarrow 0$;
        end
        else if $M[u][v] == 1$ **then**
          $M[u][v] \leftarrow 0$; $M[v][u] \leftarrow 0$;
        end
      end
      Selection_Paths$(v, t_i, L_t + E[u][v], \beta_i)$;
      Recovery site;
    end
  end
end

In addition, we also use another weighted value matrix E to describe the latency of the links, where each element $E[u][v]$ in E denotes the latency of link $e(u, v)$ as $l_{i(u,v)}$.

We then employ a recursive traversal algorithm called Depth First Search Algorithm (DFS) to select the feasible paths. Specifically, we first choose one of the AP $d_j \in \phi_i$ and add the queuing delay on AP d_j to the sum of the latency. Considering the current server is u, for each available link $e(u, v)$, we add it to the path as long as the sum of the latency is still under limitation. If link $e(u, v)$ is successfully added to the path, we then should update the adjacency matrix M accordingly. In the case of $M[u][v] = 1$, i.e., the link $e(u, v)$ is undirected and can only be traversed once, we update $M[u][v] = 0$ and $M[v][u] = 0$, which means the $e(u, v)$ and $e(v, u)$ will be removed from graph $\mathbb{H}$. In another case of $M[u][v] = 2$,

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

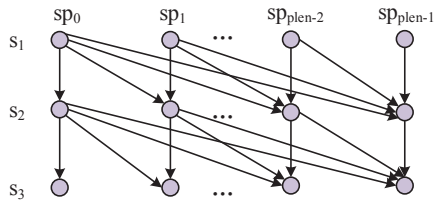


Fig. 4. The service chain $S_i = \{s_1, s_2, s_3\}$ is routed on a pre-routing path $p = \{sp_0, sp_1, ..., sp_{plen-1}\}$. The arrows from VNF s_i to server node sp_j denote node sp_j is the next feasible hop for VNF s_{i+1}.

we remove directed link $e(u, v)$ from the graph by setting $M[u][v] = 0$, and keep the directed link $e(v, u)$ unchanged. By recursively calling the function **Selection_Paths** in Algorithm 1 until reaching node t_i, we can obtain the set of pre-routing paths P_i^Λ for S_i. In the end, if there does not exist any feasible paths from $d_j \in \phi_i$ to t_i, the service chain S_i will be rejected.

D. Assigning VNFs

Given a single service chain S_i and a set of pre-routing paths P_i^Λ, our next objective is to choose an assignment scheme with the minimum resource consumption. Since we cannot foresee which path will yield the optimal solution, we take turns to pick the pre-routing path p out of P_i^Λ and compute a local optimal assignment scheme for each p. Finally, by comparing all the local optimal schemes, we choose the globally optimal scheme among all the pre-routing paths.

For a selected pre-routing path $p \in P_i^\Lambda$, let k be the length (the number of links) of the path, the corresponding number of servers along the path is $k + 1$. Hence, we can define path p as a sequence of server nodes $p = \{sp_0, sp_1, ..., sp_k\}$ ($sp_i \in \mathbb{D}, i \in \{0, 1, , ..., k\}$). Since there might loop in path p, different server nodes sp_k, $sp_{k'} \in p$ might correspond to the same server $d_j \in \mathbb{D}$. For each server $d_j \in \mathbb{D}$, we should make sure that the resource consumption of all VNFs in d_j should not exceed its resource capacity, and the processing capacities of all VNFs in d_j have sufficient capacities to process the flows. Besides, it is essential to preserve the ordered sequence of the VNFs in the service chain. Unfortunately, the assignment subproblem can be reduced as a bin-packing problem, which is still an NP-hard problem.

A Path-based Greedy Algorithm (PGA). To solve the problem, we propose a near-optimal algorithm called PGA. The key idea of PGA is to sequentially assign VNFs of S_i to the nodes along p while reusing as many available VNF instances as possible, with the constraints of the server resources and VNF processing capacities.

To begin with, we explain how to compute the maximum number of reusable VNF instances. Given a service chain $S_i = \{s_0, s_1, ..., s_{|S_i|-1}\}$ and a pre-routing path $p = \{sp_0, sp_1, ..., sp_k\}$ as inputs, we first construct a new $|S_i| \times (k + 1)$ matrix A to describe the relationship between S_i and p. Each element $A[l][j] \in A$ indicates whether $s_l \in S_i$ could reuse the VNF instance at server node sp_j. We denote by $W_{j,n}$ the residual type-n VNF instance processing capacity of device d_j, which can be obtained from Sec. III. Assuming the

Algorithm 2: A Path-based Greedy Algorithm (PGA)

Input: service chain S_i, flow rate r_i, pre-routing path p (with a length k), W_{jn}, C_j
Output: paths vector B, server consumption C_s, reusing number p_r
Initialize: $l = 0$, $j = 0$, $C_s = 0$, $p_r = 0$, $B = \emptyset$
Find the corresponding VNF type-n_l of each $s_l \in S_i$ and the resource consumption of a type-n_l instance denoted as R_l^i;
Find the corresponding server d_j of each $sn_j \in p$, the residual resource capacity C_j, and the residual processing capacity of all the VNF instances.
Compute matrix A, according to Eq. (11).
function assignment(l, j, C_s, p_r)
  for $y = j \to k$ **do**
    Denote $flag$ by whether there's a type-N server for s_l;
    if *d_y is type-N for s_l ($A[l][y] == 0$ and $C_y \geq R_l^i$) and* $flag == 0$ **then**
      $flag \leftarrow 1$;
      Resource consumption: s_l: $C_s^1 \leftarrow C_s + R_l^i$;
      Update the reusing number: $p_r^1 \leftarrow p_r$;
      if *There exists VNF has not been assigned* **then**
        Call assignment$(l + 1, y, C_s^1, p_r^1)$;
      end
      Update resource capacity of servers in p;
      $flag \leftarrow 1$;
    end
    if *Server d_y is type-N for s_l ($A[l][y] == 1$)* **then**
      Resource consumption remains unchanged: $C_s^2 \leftarrow C_s$;
      Update the reusing number: $p_r^2 \leftarrow p_r + 1$;
      if *There exists VNF has not been assigned* **then**
        Call assignment$(l + 1, y, C_s^2, p_r^2)$;
      end
      break;
    end
    Record y to the deployment vector B;
  end
  Choose the assignment scheme with the minimum C_s
end

s_l is type-n_l and the server node sp_j corresponds to server d_j, then $A[l][j]$ can be calculated as follows:

$$A[l][j] = \begin{cases} 1 & \text{if } W_{(d_j),(n_l)} - r_i \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

For example, for the flow of Fig. 3(a) and the VNF capacity as shown in Fig. 3(b), we have:

$$A = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} d_0 & d_1 & d_2 & d_3 & d_2 & d_4 & d_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}. \tag{12}$$

Besides, if VNF $s_l \in S_i$ requires a new instance, we assume its resource consumption is R_l^i.

It is vital to keep the order of the VNFs. For example, if we deploy $s_l \in S_i$ to sp_j, then $s_l' \in S_i (l' > l)$ can only be deployed at $sp_j' (j' \geq j)$, as shown in Fig. 4. We employ a recursive function **assignment** in Algorithm 2 to illustrate the operations in detail. For each $s_l \in S_i$, there are two types of servers where s_l can be assigned: the servers with reusable VNF instance denoted as type-R and the server without reusable VNF instance but with enough resource to launch at least a new VNF instance denoted as type-V. And there are four cases in each level of the recursion: 1) we first find the type-R and terminate the loop for s_l, this would be

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

the best assignment position for s_l; 2) we first find the type-V and then the type-R in sequence; 3) we only find the type-V; 4) we find neither types, i.e., nowhere s_l can be assigned, thus service chain S_i will be rejected. Note that for each level of the recursion, we only need to try at most two optional nodes for s_l assignment. Hence, for each s_l, we call the recursive function up to two times, which significantly reduces the computation complexity of PGA.

For each pre-routing path $p \in P_i^\Lambda$, we can only get a local optimal assignment solution. To get the globally optimal solution, we first compute the total resource consumption of p by adding the minimum server resource consumption and the bandwidth consumption (the product of the length of p and flow rate of S_i) together. We then compare the total resource consumption of each pre-routing path and choose the globally optimal solution as our final result. In this way, the solution obtained by the two-stage strategy is still optimal.

E. The Worst-Case Performance Bound

However, the optimal solution obtained by PGA is not optimal for the VNF chain deployment problem, since we aim at finding a deployment scheme with maximum reusing number and then calculating the minimum resource consumption, rather than directly deriving minimum resource consumption of the problem. For example, for $S_i = \{s_1, s_2, s_3, s_4\}$, we get an "optimal" solution A through PGA: if there are only reusable VNF instances for s_1, s_2 and s_3, then the maximum reusing number will be 3 and the server resource consumption will be R_4^i. In another case, say there's a solution B, where s_3, s_4 can reuse the existing VNFs. Then the reusing number of B is 2, and the server resource consumption is $R_1^i + R_2^i$. In the case of $R_1^i + R_2^i < R_4^i$, solution B is superior to our PGA solution A. That is to say, we obtain a near-optimal solution for all the pre-routing paths in P_i^Λ by PGA. Accordingly, our two-stage algorithm is near-optimal. Now, we compute the worst-case performance bound of the two-stage solution in detail.

Specifically, given a service chain S_i and a pre-routing path $p \in P_i^\Lambda$, we have obtained the optimal assignment scheme with maximum reusing number p_r and minimum server resource consumption C_s. Then the minimum total resource consumption C among all the pre-routing paths $p \in P_i^\Lambda$ (with a length denoted as k) can be calculated as:

$$C = \min\{C_s + r_i \cdot k\}, \forall p \in P_i^\Lambda. \tag{13}$$

Lemma 1. *For a service chain S_i and a same pre-routing path, assume that the reusing number of our solution and the optimal solution are p_r^A and $p_r^O (p_r^A, p_r^O \in \{1, 2, ...\})$, respectively. If our two-stage solution is not optimal, then we can get that:*

$$2 \le p_r^A \le |S_i| - 1 \quad and \quad 1 \le p_r^O \le p_r^A - 1. \tag{14}$$

Proof. 1) Firstly, we can assert that p_r^A is the maximum reusing number according to Algorithm 2, so $p_r^O \le p_r^A$. 2) If $p_r^A = p_r^O$, according to Algorithm 2, our two-stage solution will be optimal, which is contrary to the assumption. So we have $p_r^O \le p_r^A - 1$. 3) If $p_r^A = |S_i|$, which means all the VNF of S_i could reuse the existing VNF instances, then our two-stage solution will be optimal. So $p_r^A \le |S_i| - 1$. 4) If $p_r^A = 1$, then $p_r^A > p_r^O = 0$, i.e., there's no VNF could reuse the existing VNF instances in the optimal solution. Thus our two-stage solution is superior to the optimal solution, which also contradicts the assumption. So we have $p_r^A \ge 2$. All in all, we prove that $2 \le p_r^A \le |S_i| - 1$ and $1 \le p_r^O \le p_r^A - 1$. □

Theorem 1. *For service chain S_i, we define $R_{max}^i = \max\{R_1^i, R_2^i, ..., R_{|S_i|}^i\}$, $R_{min}^i = \min\{R_1^i, R_2^i, ..., R_{|S_i|}^i\}$ and $\gamma_i = R_{max}^i / R_{min}^i$ ($\gamma_i \ge 1$). Assume that the server resource consumption and the length of the selected pre-routing path of our two-stage solution and the optimal solution are C_s^A and k^A, and C_s^O and k^O, respectively. The corresponding resource consumption of the two solutions are $C^A = C_s^A + r_i \cdot k^A$ and $C^O = C_s^O + r_i \cdot k^O$ ($C^A \ge C^O$), respectively. Then, the worst-case performance bound f satisfies:*

$$f_i = \frac{C^A}{C^O} \le \gamma_i. \tag{15}$$

Proof. We proof Theorem 1 in three cases:

Case 1: If our two-stage solution is optimal, then $C^A = C^O$, we have $f_i = 1 \le \gamma_i$.

Case 2: If our two-stage solution and the optimal solution are different ($C^A > C^O$) and yield from the same pre-routing path p_k whose length is k. Then the bandwidth consumption is the same for our two-stage solution and the optimal solution: $r_i \cdot k^A = r_i \cdot k^O = r_i \cdot k$. Hence, we have $C_s^A > C_s^O$. By applying Lemma 1, we have $p_r^O \le p_r^A - 1$. Therefore, the server resource consumption of our two-stage solution and the optimal solution are $C_s^A = \overbrace{R_x^i + ... + R_y^i}^{|S_i| - p_r^A}$ and $C_s^O = \overbrace{R_{x'}^i + ... + R_{y'}^i}^{|S_i| - p_r^O}$, respectively. Furthermore, since $C_s^A > C_s^O$ and $r_i \cdot k^A = r_i \cdot k^O$, we have

$$f_i = \frac{C^A}{C^O} = \frac{C_s^A + r_i \cdot k^A}{C_s^O + r_i \cdot k^O} < \frac{C_s^A}{C_s^O} = \frac{\overbrace{R_x^i + ... + R_y^i}^{|S_i| - p_r^A}}{\underbrace{R_{x'}^i + ... + R_{y'}^i}_{|S_i| - p_r^O}}. \tag{16}$$

The inequality (16) holds since we can easily obtain that $\frac{b+c}{a+c} \le \frac{b}{a}$, $\forall a, b, c > 0$ and $b > a$.

In order to get the maximum f, we calculate the maximum value of the numerator and the minimum value of the denominator respectively. Hence, we have:

$$\begin{aligned} f_i &= \frac{\overbrace{R_x^i + ... + R_y^i}^{|S_i| - p_r^A}}{\underbrace{R_{x'}^i + ... + R_{y'}^i}_{|S_i| - p_r^O}} \le \frac{\overbrace{R_x^i + ... + R_y^i}^{|S_i| - p_r^A}}{\underbrace{R_{x''}^i + ... + R_{y''}^i}_{|S_i| + 1 - p_r^A}} \\ &\le \frac{R_{max}^i (|S_i| - p_r^A)}{R_{min}^i (|S_i| + 1 - p_r^A)} = \gamma_i (1 - \frac{1}{|S_i| + 1 - p_r^A}) \\ &< \gamma_i. \end{aligned} \tag{17}$$

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

Case 3: If our two-stage solution and the optimal solution are different ($C^A > C^O$) and yield from different pre-routing paths, let's say p^A and p^O respectively. For path p^O, we define the total resource consumption yields from our two-stage solution is C^{O1}. Since C^{O1} and C^A are both from our solution, it is known that $C^A \leq C^{O1}$. Also because C^O is the optimal solution, then we have $C^O \leq C^A \leq C^{O1}$. It has been proven that $f_i = \frac{C^{O1}}{C^O} < \gamma_i$ in Case 2, hence, $\frac{C^A}{C^O} \leq \frac{C^{O1}}{C^O} < \gamma_i$.

All in all, for service chain S_i, we have proven that the worst-case performance bound is $f_i = \frac{C^A}{C^O} \leq \gamma_i$. □

Corollary 1. *Represent the worst-case performance bound F of all the arriving service chain $S_1, S_2 ... S_N$ and assume $\gamma^k_{max} = \frac{R^k_{max}}{R^k_{min}} = \max\{\gamma_1, \gamma_2 ..., \gamma_N\}$, then we can derive F as follows.*

$$F = \frac{\sum_{i=1}^{N} C_i^A}{\sum_{i=1}^{N} C_i^O} \leq \frac{\sum_{i=1}^{N} \gamma_i \cdot C_i^O}{\sum_{i=1}^{N} C_i^O} \leq \frac{\gamma^k_{max} \cdot \sum_{i=1}^{N} C_i^O}{\sum_{i=1}^{N} C_i^O} = \gamma^k_{max}. \quad (18)$$

The inequality (18) holds since $C_i^A \leq \gamma_i \cdot C_i^O$ according to Theorem 1.

F. Complexity Analysis

Now we give a brief complexity analysis of our proposed algorithm. In Algorithm 1, without regard to branch reduction, finding the feasible paths in the N-node network requires $O(N!)$ (from line 3-27) computation. In the second Algorithm 2, assigning a service chain with M nodes along a path requires $O(2^M)$ (line 5-29). Then the time complexity of the two-stage algorithm is $O(N! \cdot 2^M)$. However, since at the realistic network edge, to meet the low-latency requirement and the bandwidth constraints of the physical links, the computation complexity of the first algorithm will be significantly reduced. Besides, the length of the service chain is less than 10 [36] in the vast majority of cases, the computation complexity in Algorithm 2 is not large. We also conduct simulations to prove the two-stage algorithm is quite efficient with realistic network topologies, the results of the execution time is depicted as Fig. 5 and Fig. 6 in Sec. V.

V. EVALUATION

In this section, we conduct simulations to evaluate the performance of our proposed algorithm on top of the real-world network topologies and generate other parameters according to some previous works [37], [26].

Network topology. We implement the experiments on several network topologies of different sizes from the Internet topology zoo [38]: such as Bellsouth (51 nodes and 66 links), Cogentco (197 nodes and 245 links) and Kdl (754 nodes and 899 links) to simulate three different network scales. According to [37], we configure the nodes in the network with [0, 200] units of residual available resource capacities. To characterize the current configuration of each server, we randomly generate a set of reusable VNFs consisting of 0 to 8 VNFs picked from 20 VNF types. We further assign a random bandwidth capacity ranging from 0 to 1000 Mbps to each link between the servers.

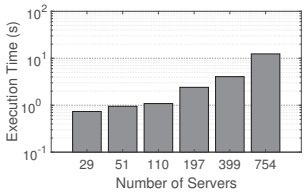


Fig. 5. The execution time with different number of servers.

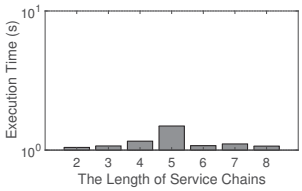


Fig. 6. The execution time with different lengths of service chains.

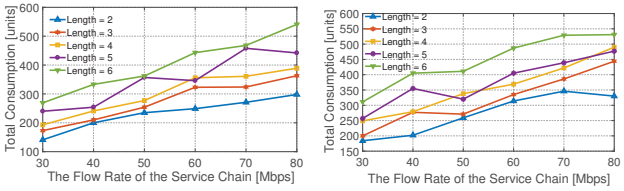
Service chains. As aforementioned in Sec. IV, due to the assumption that the demand of service arrives into the system one by one, we build a single service chain at a time with a set of APs (source nodes) and a single destination randomly chosen from the set of nodes of the network topology. Specifically, we configure each AP with a processing capacity ranging from 100 Mbps to 200 Mbps. In addition, we generate a service chain by randomly picking 1 to 6 VNFs from 20 VNF types, and each type of VNF instance requires [20, 50] units of server resources in terms of CPU and memory. The flow rate of the chain ranges from 30 Mbps to 60 Mbps and the deadline are randomly distributed in [30, 80] ms [26]. We repeat each experiment 20 times to eliminate contingency.

Algorithm benchmark. In the simulations, we compare our proposed algorithm with the other three heuristic algorithms. Since our algorithm consists of two parts, i.e., constrained depth-first search algorithm (CDFSA) and the path-based greedy algorithm (PGA), we use these two parts to combine with some previous heuristics, (e.g., shortest path heuristic and first fit heuristic) to generate the other three algorithms, the details are as follows.

- **Shortest Path Heuristic + First Fit Assignment (SHP+FF)**: in the first stage, it finds the shortest path by shortest path heuristic (SPH), and in the second stage, it assigns the VNFs along the shortest path by the first fit algorithm.
- **Constrained Depth-First Search Algorithm + First Fit Assignment (CDFSA + FF)**: it selects the feasible paths with CDFSA, and employs first fit (FF) algorithm to assign the VNFs along the path. In the end, it will calculate the optimal solution among all the chosen paths.
- **Shortest Path Heuristic + Path-based Greedy Algorithm (SHP+PGA)**: it employs the SPH to select the shortest path, and uses PGA to assign the VNFs.

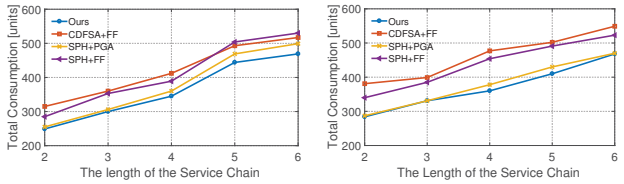
Execution time. From the previous analysis, the time complexity of our proposed algorithm is correlated with the number of the servers and the length of the service chains. We first conduct the experiments in six realistic networks topologies with different scales. As shown in Fig. 5, as the number of servers ranges from 29 to 754, the execution time increases accordingly. Specifically, in the small-scale network with less than 110 servers, it takes less than 1 s to get a solution for 100 service chains. In the large-scale network, such as the 754-node case, the execution time is 12.39 s, which is still acceptable. Fig. 6 shows how the length of

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.



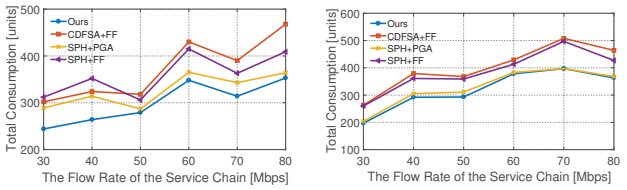
(a) Bellsouth with 51 nodes (b) Cogentco with 197 nodes

Fig. 7. Impact relation between the length and the rate of service chains.



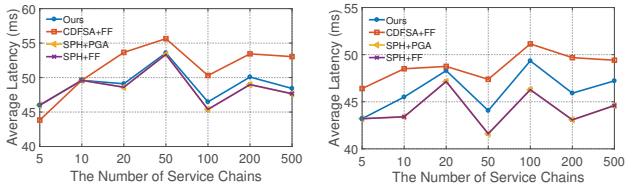
(a) Bellsouth with 51 nodes (b) Cogentco with 197 nodes

Fig. 8. Total resource consumption with different lengths of service chains.



(a) Bellsouth with 51 nodes (b) Cogentco with 197 nodes

Fig. 9. Total resource consumption with different flow rates of service chains.



(a) Bellsouth with 51 nodes (b) Cogentco with 197 nodes

Fig. 10. The average latency with different number of service chains.

the service chains affects the execution time. The simulation is conducted with a 197-node network (Cogentco) topology and 100 service chains at one time. The simulation results demonstrate that the execution time does not vary linearly with the length, probably because the longer the service chain is, the fewer feasible paths can be obtained in the first stage. Even when the length is 5, the maximum execution time for 100 service chains is just 2.56 s. Hence, although the time complexity is theoretically exponential, our proposed algorithms can efficiently get a feasible solution even in quite large-scale networks considering the characteristics of the realistic topologies and the service chains.

The total resource consumption. Since our objective in Eq. (9) is to jointly minimize the server resource and bandwidth resource, similar to [18], we mainly consider the length of the service chains which affects the consumption of server resource and the flow rate of the service chain which affects the consumption of the bandwidth resource. First of all, in Fig. 7, we depict the impact relationship between the length and the rate of the service chain with network Bellsouth (51 nodes) and Cogentco (197 nodes), respectively. As expected, the total resource consumption not only increases with the length, but also increases with the flow rate of the service chains.

Next, we explore the total resource consumption of our proposed algorithm comparing with other heuristic algorithms with different lengths and flow rates of services chains, respectively, as shown in Fig. 8 and Fig. 9. It is obvious that the total resource consumption of all algorithms increases with the length and flow rate of the service chain. Compared with other algorithms, our proposed algorithm can consume averagely 19.7%, 8.9% and 23.9% less resource than SPH+FF, SPH+PGA, and CDFSA+FF on network Bellsouth, and correspondingly 20.7%, 2.5% and 25.6% on Network Cogentco. The superiority comes from the two-stage design of our algorithms. On one hand, in the first stage, unlike the shortest path heuristic, we select all the feasible paths to make a global decision. On the other hand, we assign the VNFs along the paths using PGA to reuse as many VNFs as possible, which outperforms the first fit algorithm.

It is worth noting that the result of the algorithm SPH+PGA is almost close to ours, probably because the path of the optimal scheme of our algorithm is usually the shortest path in the network. In addition, based on the comparison between each subfigure pair in Fig. 7, Fig. 8 and Fig. 9, we can find that when the length and the flow rate are the same, the total resource consumption on network Cogentco is almost larger than that of network Bellsouth. Probably because we randomly generate the source and destination for a service chain, the larger the network is, the longer the feasible path might be.

Average latency. We now depict the average latency by comparing different algorithms. As shown in Fig. 10(a) and Fig. 10(b), the average latency of service chains ranges from 40 ms to 55 ms. Note that the SPH+FF and SPH+PGA always keep the same pace and incur minimum average latency, because they both choose the shortest path to deploy service chains. However, as shown in Fig. 8 and Fig. 9, the other three algorithms consume more resources than our algorithm. Also, even our algorithm is not the best one in terms of latency, it can also meet the latency requirements of the service chains in the network edge.

VI. CONCLUSION

In this paper, we investigate the VNF chain deployment problem at the network edge, to simultaneously reuse the server resources and reduce the bandwidth consumption, while providing the latency guarantees. We first reveal the dilemma between the resource utilization of servers and links. Then we formalize the problem as an MILP model and devise a two-stage solution to solve it. In the first stage, we employ CDFSA to select all the pre-routing paths. Given these paths, in the second stage, we design PGA to assign VNFs with minimum resource consumption. Our solution is proven to be near-optimal with a theoretically proved worst-case performance bound. Finally, we conduct simulations on realistic network topologies. The results demonstrate our proposed algorithm outperforms three other heuristic algorithms.

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] D. Ma, X. Fan, J. Gausemeier, and M. Grafe, *Virtual reality & augmented reality in industry*. Springer, 2011.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] S. Wang, K. Chan, R. Urgaonkar, T. He, and K. K. Leung, "Emulation-based study of dynamic service placement in mobile micro-clouds," in *IEEE Military Communications Conference, (MILCOM)*, 2015, pp. 1046–1051.

[4] R. Cziva and D. P. Pezaros, "Container network functions: bringing nfv to the network edge," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 24–31, 2017.

[5] G. Fettweis, H. Boche, T. Wiegand, E. Zielinski, H. Schotten, P. Merz, S. Hirche, A. Festag, W. Häffner, M. Meyer *et al.*, "The tactile internet-itu-t technology watch report," *Int. Telecom. Union (ITU), Geneva*, 2014.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[7] S. I. Vesterinen, M. M. Aalto, S. J. Kekki, and J. I. Hongisto, "Method, radio system, mobile terminal and base station for providing local breakout service," Jun. 11 2013, uS Patent 8,462,696.

[8] A. T. Chow, R. H. Erving, J. Kim, R. M. I. Robert, J. E. Russell, and W. Ying, "Computer readable medium with embedded instructions for providing communication services between a broadband network and an enterprise wireless communication platform within a residential or business environment," Apr. 10 2012, uS Patent 8,155,155.

[9] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "Mobiscud: A fast moving personal cloud in the mobile network," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM, 2015, pp. 19–24.

[10] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[11] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1040–1050.

[12] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *IEEE INFOCOM*, 2019, pp. 2566–2574.

[13] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, 2009.

[14] M. Abrash, "Latency the sine qua non of ar and vr," *Blog post, Dec*, 2012.

[15] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *IEEE INFOCOM*, 2016, pp. 1–9.

[16] Z. Zhang, Z. Li, C. Wu, and C. Huang, "A scalable and distributed approach for nfv service chain cost minimization," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2151–2156.

[17] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *International Conference on Network and Service Management (CNSM)*, 2015, pp. 50–56.

[18] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Transactions on Networking*, no. 99, pp. 1–15, 2018.

[19] 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 International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 731–741.

[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] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *IEEE INFOCOM*, 2015, pp. 1346–1354.

[22] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "Nfvdeep: adaptive online service function chain deployment with deep reinforcement learning," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[23] 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.

[24] 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.

[25] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE INFOCOM*, 2019, pp. 1459–1467.

[26] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware vnf placement for service-customized 5g network slices," in *IEEE INFOCOM*, 2019, pp. 2449–2457.

[27] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," 2018.

[28] W. Ma, O. Sandoval, J. Beltran, D. Pan *et al.*, "Traffic aware placement of interdependent nfv middleboxes," in *IEEE INFOCOM*, 2017, pp. 1–9.

[29] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014, pp. 7–13.

[30] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[31] P.-W. Chi, Y.-C. Huang, and C.-L. Lei, "Efficient nfv deployment in data center networks," in *IEEE International Conference on Communications (ICC)*, 2015, pp. 5290–5295.

[32] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De 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, 2015.

[33] A. Lodi, S. Martello, and D. Vigo, "Approximation algorithms for the oriented two-dimensional bin packing problem," *European Journal of Operational Research*, vol. 112, no. 1, pp. 158–166, 1999.

[34] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proceedings IEEE International Conference on Robotics and Automation*, 1992, pp. 1186–1192.

[35] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[36] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[37] S. Khebbache, M. Hadji, and D. Zeghlache, "Scalable and cost-efficient algorithms for vnf chaining and placement problem," in *IEEE Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 92–99.

[38] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

Authorized licensed use limited to: Telecom ParisTech. Downloaded on October 11,2023 at 08:44:43 UTC from IEEE Xplore. Restrictions apply.