Provable Approximation Algorithms for Online Traffic-Sensitive SFC Deployment

Yingling Mao, Xiaojun Shang, and Yuanyuan Yang, *Life Fellow, IEEE*

***Abstract*—Network Function Virtualization (NFV) has the potential for cost-efficiency, manage-convenience, and flexibility services but meanwhile poses challenges for the service function chain (SFC) deployment problem, which is NP-hard. It is so complicated that existing work conspicuously neglects the flow changes along the chains and only gives heuristic algorithms without a performance guarantee. In this paper, we fill this gap by formulating a traffic-sensitive online joint SFC placement and flow routing (TO-JPR) model, with the objective of jointly optimize the resource cost and network latency, and proposing a novel two-stage scheme to solve it. We design a dynamic segmental packing (DSP) algorithm for the first stage, which not only maintains the minimal traffic burden for the network but also achieves an approximation ratio of a small constant on the resource cost. Besides, we propose the greedy mapping (GM) algorithm for the second stage, which can guarantee a global approximation ratio of O(d) on the network latency. Here d is the diameter of the network graph and is typically smaller than O(log(M)), where M is the number of servers in the network. Finally, we perform extensive simulations to demonstrate the outstanding performance of our algorithms compared with the optimal solutions and benchmarks.**

***Index Terms*—Network function virtualization, service function chain placement, flow routing, resource optimization, network latency optimization.**

I. Introduction

NETWORK function virtualization [1] is penetrating our daily life. It makes network services more cost-efficient, manage-convenient, and flexible, by migrating network functions, or middleboxes, from proprietary hardware appliances to common commercial servers. Specifically, the various network services are now realized by chained-up virtual network functions (VNFs), also called service function chains (SFCs) [2]. But the flexibility of VNFs also poses challenges to the deployment problem of SFCs. Elegant SFC deployment can further improve both efficiency and cost-effectiveness of network services, which fully utilize computational resources without increasing too much network latency. On the contrary, improper deployment decisions may cause redundant flow paths and traffic burdens, and thus incur unacceptable network latency and even network congestion.

SFC deployment attracts great attention from academia. A variety of related work, e.g., [5], [6], [7], [8], [9], [10] and [11], has elaborated on deploying SFC efficiently, which is typically formulated into a Joint SFC Placement and flow Routing problem (JPR). However, due to the NP-hardness of JPR, most of the existing solutions, e.g., [5], [6], [7] and [8], are limited to the design of heuristic algorithms and do not have a provable performance guarantee. To the best of our knowledge, there are only three existing works [9], [10], [11] having the performance bound. But the achieved bounds are either loose or impractical, seen in detail from the Related Work (Sec. II). In our model, we pursue a near-optimal deployment scheme that saves resource utilization and reduces network latency at the same time and maintains a much tighter performance bound compared with the existing works.

In addition, as far as we are concerned, most models in previous work, including those with a performance guarantee, ignore the flow changes between VNFs on each SFC. In practice, many VNFs have an influence on the traffic volumes of the processed data flows. For example, the Citrix CloudBridge WAN optimizer compresses the data flow, reducing the traffic volume by up to 80%. The BCH(63,48) encoder increases the traffic volume by 31% due to the checksum overhead [12]. According to Little's law or the popular Cisco EIGRP [25] protocols, the traffic volume has a close relationship with the produced network latency. In particular, the larger the traffic volume, the higher the produced link latency. Thus, it is necessary to take such flow changes into consideration in JPR, and such consideration makes the problem even more complicated. To the best of our knowledge, there are two existing works [12], [13] considering the flow changes and it is also limited to the design of heuristic algorithms with no performance guarantee.

In this paper, we formulate a joint SFC placement and flow routing model which considers the flow changes along each service chain and deals with sequential arrivals and arbitrary leaves of SFCs in an online manner. Based on the model, we propose a traffic-sensitive online SFC placement and flow routing (TO-JPR) scheme. In our scheme, we creatively put forward a new conception called *traffic burden*, i.e., the total link latency of the data flow which needs to be transmitted via the network. It helps break down the TO-JPR problem

Received 4 April 2023; revised 3 July 2024 and 4 February 2025; accepted 22 April 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor S. Moharir. This work was supported in part by U.S. National Science Foundation under Grant CCF-1717731 and Grant CCF-2230620. *(Corresponding author: Yuanyuan Yang.)*

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

Xiaojun Shang is with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76019 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TON.2025.3566728, provided by the authors.

Digital Object Identifier 10.1109/TON.2025.3566728

and make it easier to solve. Specifically, with traffic burden as a pivot, our scheme has two stages. In the first stage of the scheme, our task is to pack VNFs, with the aim of minimizing the resource cost and the produced traffic burden. We call it the packing stage. And in the second stage, named the mapping-and-routing stage, we focus on mapping the VNF packages onto the real servers one by one and routing the data flow between the VNF packages.

The two-stage scheme has lots of advantages. First, it wisely split the two optimization objectives into two stages, making TO-JPR easier to solve. Besides, since we can perfectly minimize the traffic burden in the packing stage as shown in Thm. 2, the performance guarantee of the mapping-and-routing stage can become the global performance guarantee for the network latency in the whole SFC deployment problem. Additionally, a such split measure based on the concept of traffic burden is also practical. Considering advanced datacenter structures [3], [4], commercial servers can retrieve data of other servers from their shared memory. With such technologies, the impact of data routing is lifted. In this way, the mapping-and-routing stage can be ignored and the packing stage, reducing the resource cost meanwhile minimizing the traffic burden, is the whole story. Moreover, the two-stage scheme has strong expansibility. For example, some scenarios, e.g., [14], prefer balanced network flow rather than low network latency. In these situations, our two-stage scheme still works by merely changing the optimization objective of the mapping-and-routing stage, and our outstanding theoretical results of the packing stage are preserved.

As for the packing stage, we put forward a dynamic segmental packing (DSP) algorithm, which sequentially packs each SFC according to its arriving order. DSP solves the conflict between two optimization goals. It not only perfectly solves the optimization task of the traffic burden, but also achieves a resource approximation ratio of $2+\frac{1}{\gamma}$, where γ is the minimum ratio of totally needed computing resource of each SFC to the size of servers. Additionally, as for some special online cases, where SFCs queue to leave in their arriving order, QSP can further reach an asymptotic approximation ratio of 2 on the resource cost. When handling the mapping-and-routing stage, we propose a greedy mapping (GM) algorithm, which can deal with arbitrary leaves of SFCs and cooperate with DSP to obtain a global ratio of $O(d_G)$ on the network latency, where d_G is the diameter of the network graph. In common practical network topologies, d_G is typically smaller than $O(log(M))$, where M is the number of servers in the network.

Our main contributions are listed as follows.

- We formulate a joint online SFC placement and flow routing model which considers the flow changes along each service chain and deals with sequential arrivals and arbitrary leaves of SFCs in an online manner, while optimizing both resource cost and network latency.
- We creatively propose a two-stage traffic-sensitive online SFC placement and routing scheme. In the first stage, we aim to minimize the resource cost and the traffic burden by packing VNFs, while in the second stage, we target optimizing the network latency based on the produced traffic burden.
- We put forward a dynamic segmental packing (DSP) algorithm for the first stage, which not only maintains the minimal traffic burden but also achieves a resource approximation ratio of a small constant.
- As for the second stage, we propose a greedy mapping (GM) algorithm in the general online case, which guarantees a global ratio of $O(d_G)$ on the network latency. Here d_G is the diameter of the network graph and typically smaller than $O(log(M))$, where M is the number of servers in the network.
- We perform extensive simulations to evaluate and compare the performance of our proposed algorithms with those of OPTs and benchmarks.

The remainder of this paper is organized as follows. Section II reviews the related works. In Section III, we describe the system model, formulate the TO-JPR problem, and prove its NP-hardness. Section IV proposes a two-stage scheme to solve TO-JPR. Afterward, Section V introduces DSP for the first stage while Section VI gives GM for the second stage. Then Section VIII is the performance evaluation of DSP+GM. Finally, we conclude the paper in Section IX.

II. Related Work

In order to fully launch its potential of cost-efficiency and flexibility, the SFC deployment problem has become a research hot spot in NFV. A variety of research [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] have been devoted to it, targeting different optimization goals. Among these, the two most popular and influential objectives are resource consumption and network latency. For instance, researches [15], [16], [17], [18] all contribute to resource optimization while [19], [20], [21] pursue the lowest latency.

Although the two objectives may conflict, making it more challenging for the SFC deployment problem, they are both vital in practice. Thus, it is necessary for us to jointly consider resource consumption and network latency in our model. Work [5], [6], [7], [8], [9], [10], [11] have done so, but most of the approaches, e.g., [5], [6], [7] and [8], are heuristic algorithms with no provable performance guarantee. To the best of our knowledge, there are only three existing works [9], [10], [11] that consider resource consumption and network latency having the performance bound. Work [9], [10] put forward provable approximation algorithms based on the rounding algorithm. In [9], the algorithm achieves a total approximation factor of 8, but it may violate servers' capacities by a factor of 16. Such a violation factor is so great that it may cause network delay, or even crash. In [10], the capacity limitation is satisfied with an approximation ratio of $O(log(M))$, where M is the number of servers. Such a ratio is relative to M, so it will be dramatically big in the large-scale case. Reference [11] designs a two-stage VNF deployment scheme with a constrained depth-first search algorithm (CDFSA) and a path-based greedy algorithm (PGA). They give a theoretically-proved worst-case performance bound by a constant factor. But such a factor is implicit and may be very large.

Besides, the flow change is also critical for consideration. Although the traffic-sensitive JPR is not novel, it is creative to

TABLE I
NOTATIONS

M	number of physical servers in the network
r	number of routers (or exchange points) in the network
V_k	physical server k
C	unit capacity of commercial servers
A	activation cost (the energy and capital cost) for a server
$B_{p,q}$	bandwidth limit of network connection (p, q)
m	total number of Service Function Chains (SFCs)
T	the entire time span for the TO-JPR problem
t_i^a	the arrive time of the request for SFC i
t_i	the existing time of SFC i in system
N	number of all VNFs
n_i	number of VNFs in SFC i
$F_{i,j}$	VNF j in SFC i
$f_{i,j}$	needed processing capacity of VNF $F_{i,j}$
$b_{i,j}$	traffic volume of data flow after passing $F_{i,j}$
$l_{i,j}$	link latency of data flow after passing $F_{i,j}$
$x_{i,j}^k$	$= 1$, iff VNF $F_{i,j}$ is placed on V_k
$w_{i,j}^{p,q}$	$= 1$, iff data flow between $F_{i,j}$ and $F_{i,j+1}$ pass through link (p, q)
$e_i(t)$	$= 1$, iff SFC i exists in system or not
$y_k(t)$	$= 1$, iff server k is occupied or not at time t
$\mathbb{R}$	the total resource cost
$\mathbb{L}$	the total network latency between servers

produce a near-optimal scheme with a provable performance guarantee for this problem. Traffic-sensitive consideration makes the complicated JPR even more challenging. As far as we are concerned, there are only two existing works [12], [13] considering the flow changes and they are limited to the design of heuristic offline algorithms and do not have a provable performance guarantee.

III. PROBLEM FORMULATION

A. System Model

The notations used in this model are shown in Table I.

1) Physical Network: Consider a physical network represented by a undirected graph $G = (V, E)$, where each node is a commercial server or a router, $V = \{V_1, V_2, \cdots, V_M, \cdots V_{M+r}\}$ is the set of nodes in the network, and E is the set of communication channels connecting network nodes in V. Here, $V_1, V_2, \cdots, V_M$ are M commercial servers in the network G. Typically, commercial servers in a data center have uniform settings, specifically, CPUs, GPUs, RAM, etc. Thus, we assume that all the servers have a unit processing capacity, noted as C.[1] $V_{M+1}, \cdots, V_{M+r}$ are the routers (or exchange points) with zero computing resources.

For each pair of servers $V_p, V_q \in V$, if $(V_p, V_q) \in E$, it implies there exists a physical communication channel connecting the server V_p and server V_q. Denote its bandwidth as $B_{p,q}$ If $(V_p, V_q) \notin E$, it means server V_p and server V_q are not directly connected, where we mark $B_{p,q} = 0$.

2) SFCs: Suppose the service requests arrive sequentially in an online manner and each time a request arrives, a corresponding SFC needs to be deployed to realize such a service request. Denote SFC i as the SFC working for the i-th arriving service request. Assume there are m request arrivals in total over the entire time span T and the request for SFC i arrives at time t_i^a with a consequent existing time of t_i ($t_i^a + t_i - 1 \in T$). In our online model, the request for SFC i may leave sometime but will not arrive again after leaving, since each SFC arrival will be considered as a new SFC and indexed depending on the new arriving order. At each time slot, we do not know the information of the future, such as the information of the next arriving SFCs and the time when the arrived SFC will leave in the future. We only know the information of SFCs that have arrived and the time of the arrivals and leaves that happened. The deploying system executes the following procedures at each time slot: first removing timeout SFCs, updating the server states, receiving arriving requests, making SFC deployment decisions, and finally again updating the server states.

To ensure data security and privacy, we assign individual and personalized SFCs for each user's service request. In each SFC, VNFs are chained in a specific order according to the application requirements. Suppose there are n_i VNFs in SFC i, noted as $F_{i,1}, F_{i,2}, \cdots, F_{i,n_i}$ in chaining order. The needed computing resource of VNF $F_{i,j}$, i.e., the size of $F_{i,j}$, is noted as $f_{i,j}$. The total number of VNFs is $N = \sum_{i=1}^{m} n_i$.

There are data flows between the adjacent chained VNFs. But it's worth mentioning that different VNFs have distinct influences on the traffic volumes of the processed data flows. For example, the Citrix CloudBridge WAN optimizer compresses the data flow, reducing the traffic volume by up to 80%. The BCH(63,48) encoder increases the traffic volume of data flow by 31% due to the checksum overhead [12]. Therefore, we take the flow changes into consideration in our model and denote the traffic volume (or called flow rate) of data flow after passing VNF $F_{i,j}$ as $b_{i,j}$. Besides, we denote the communication latency of such data flow passing a network link as $l_{i,j}$, which depends on $b_{i,j}$ according to Little's law or the popular Cisco EIGRP [25] protocols. In particular, the larger the traffic volume $b_{i,j}$, the higher the produced link latency $l_{i,j}$.

3) SFC Deployment: As for each VNF $F_{i,j}$, if server V_k has enough idle capacity, no less than $f_{i,j}$, VNF $F_{i,j}$ can be placed on server V_k. If any VNF is placed on a server, we say that the server is occupied and we should pay for the energy and capital cost to run the server. If $F_{i,j}$ and $F_{i,j+1}$ are placed on the same server, data flow between $F_{i,j}$ and $F_{i,j+1}$ is not transmitted between servers so that it will not cause a traffic burden on the physic network. We call such data flow the idle data flow. If $F_{i,j}$ and $F_{i,j+1}$ are placed on different servers, the data flow between $F_{i,j}$ and $F_{i,j+1}$ needs to transmit from one server to another server, incurring traffic burden as well as network latency. We refer to such data flows as transmitting data flows.

Typically, G is not a complete graph. So the data flow from $F_{i,j}$ on one server to $F_{i,j+1}$ on another server may need to pass via several other server-nodes in G, when the two server-nodes are not directly connected. In this case, the network latency is determined jointly by the link latency of the data flow and the number of hops, i.e., network connections data flow passing through. The path of hops depends on the solutions of the

[1] We begin by addressing the TO-JPR problem under this assumption for data center scenarios. Later in the manuscript, we will relax this assumption and extend our model and proposed algorithms for more heterogeneous network scenarios, such as edge and IOT environments.

multi-hop routing problem on G under the constraints of the bandwidth limits.

In sum, our model solves a traffic-sensitive online joint SFC placement and flow routing problem (TO-JPR).

B. Problem Formulation

In the TO-JPR problem, our task is to deploy the m SFCs onto the physical network G with M commercial servers, i.e., place the N VNFs onto M servers and route the corresponding data flow in the network graph G. When doing so, there are two categories of conditions needed to be satisfied: server capacity constraints and bandwidth limitations. And our goal is to minimize the total resource consumption and the whole network latency.

In order to formulate TO-JPR, we first define two decision *Boolean* variables $x_{i,j}^k$ and $w_{i,j}^{p,q}$ as below. $x_{i,j}^k = 1$ if and only if VNF $F_{i,j}$ is placed on server k, while $w_{i,j}^{p,q} = 1$ if and only if Data Flow between VNF $F_{i,j}$ and $F_{i,j+1}$ pass through network connection (V_p, V_q).

Besides, for clear expression, we also define a *Boolean* constant $e_i(t)$ and another dependent *Boolean* variable $y_k(t)$, which is dependent on $x_{i,j}^k$. Specifically, $e_i(t) = 1$ if and only if $t_i^a \le t \le t_i^a + t_i - 1$, while $y_k(t) = 1$ if and only if server k is occupied at time t, i.e., $\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot e_i(t) > 0$.

The capacity constraint of each commercial server requires

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot f_{i,j} \cdot e_i(t) \le y_k(t) \cdot C, \quad \forall 1 \le k \le M, t \in T. \tag{1}$$

Since each VNF cannot be split, which implies it is exactly placed on a commercial server, we have

$$\sum_{k=1}^{M} x_{i,j}^k = 1, \qquad \forall 1 \le i \le m, 1 \le j \le n_i. \tag{2}$$

According to *Flow Conservation Law*, as for any data flow between VNF $F_{i,j}$ and $F_{i,j+1}$, we have $\forall\ 1 \le k \le M$,

$$\sum_{p=1}^{M+r} w_{i,j}^{p,k} - \sum_{q=1}^{M+r} w_{i,j}^{k,q} = x_{i,j+1}^k - x_{i,j}^k. \tag{3}$$

The limitation of bandwidth asks

$$sum_{i=1}^{m} \sum_{j=1}^{n_i-1} (w_{i,j}^{p,q} + w_{i,j}^{q,p}) \cdot b_{i,j} \cdot e_i(t) \le B_{p,q},$$
$$\forall\ 1 \le p, q \le M + r, \tag{4}$$

which implies the total traffic volume of all the data flow passing the network connection (p, q) should not exceed its bandwidth limitation $B_{p,q}$.

The total resource cost should contain two parts: one is the energy and capital cost to run the activated/used servers, and the other part is the used computing resource cost from the VNFs placed on the servers. Since the needed computing resources for all VNFs are known, the total used computing resource cost is identical, i.e., $\sum_{t\in T}\sum_{i=1}^{m}\sum_{j=1}^{n_i} f_{i,j} e_i(t)$, for any SFC deployment schemes. It means this part cost does not be affected by SFC deployment schemes, thus in the process of SFC deployment optimization, we omitted this part in the optimization objective function $\mathbb{R}$.

The two optimization objectives in our model are:

- The Total Resource Cost $\mathbb{R}$

$$\mathbb{R} = \sum_{t\in T}\sum_{k=1}^{M} y_k(t) \cdot A,$$

  Note that: Referring to related works [15], [16], $\mathbb{R}$ can be determined by the number of activated/used servers, which implies that even if there is only one VNF placed on the server, such server has to be activated and we should pay for its activation cost A, i.e., the energy and capital cost to run the server.

- The Network Latency between Servers $\mathbb{L}$

$$\mathbb{L} = \sum_{t\in T}\sum_{i=1}^{m} e_i(t) \cdot \left(\sum_{j=1}^{n_i-1} l_{i,j} \cdot \sum_{p=1}^{M+r}\sum_{q=1}^{M+r} w_{i,j}^{p,q} \right),$$

  where $l_{i,j}$ is the communication latency of the data flow between VNF $F_{i,j}$ and $F_{i,j+1}$ over a network link and $\sum_{p=1}^{M+r}\sum_{q=1}^{M+r} w_{i,j}^{p,q}$ represents the number of hops that data flow passes. Such modeling approach is based on [8], [11], [12] and [13].

In all, TO-JPR can be formulated as the below ILP.

$$\begin{aligned} \min \quad & \alpha\mathbb{R} + \beta\mathbb{L} \\ \text{s.t.} \quad & (1) - (4), \end{aligned}$$

where α, β are both weighting factors that are used to adjust the relative importance of the two cost components.

Note that since activation cost A and server capacity C are constant, we can redefine $\mathbb{R} = \sum_{t\in T}\sum_{k=1}^{M} y_k(t) \cdot C$ with an equivalent mathematical transformation by incorporating $\frac{A}{C}$ into α. For simplification of the proof, we will adopt this newly defined $\mathbb{R}$ in the below context.

C. Proof of NP-Hardness

In this section, we show that TO-JPR is NP-hard through a reduction from the Bin Packing (BP) Problem. It means TO-JPR cannot be solved in deterministic polynomial time unless $P = NP$.

Theorem 1: TO-JPR is NP-hard.

Proof: As for any BP problem with bin size V and item size $a_1, a_2, \cdots, a_n$, we can generate a TO-JPR problem in polynomial time by setting $M = N = n_1 = n, m = 1, f_{1,i} = a_i (1 \le i \le n), C = V, \alpha = 1, \beta = 0, B_{p,q} = \infty (1 \le p, q \le M + r), T = [t_1^a, t_1^a + t_1]$. Here, the generated TO-JPR problem and the given BP are exactly the same, thus obviously they are equivalent.

In all, BP $\le_P$ TO-JPR. Since BP is NP-hard, TO-JPR is also NP-hard. □

IV. A Two-Stage Scheme

A. Problem Complexity

The TO-JPR problem is very complex since its NP-hardness comes from not only the resource cost part but also the network latency part. The resource optimization part is shown

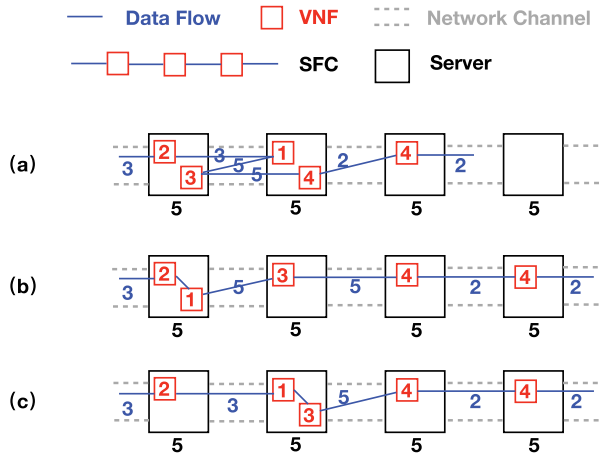


Fig. 1. An example of TO-JPR and several possible solutions.

to be NP-hard by reducing from BP in Theorem 1. Similarly, the latency optimization part of TO-JPR can be shown to be NP-hard via a reduction from the *Quadratic Assignment Problem (QAP)*.

What's worse, these two optimization objectives are not consistent. To some extent, they are conflicting. Let us take a simple case of TO-JPR for example to give readers a clear and intuitive understanding. Assume there is one SFC with 5 VNFs and 4 one-line connected servers. The link latency of data flow along the chain, the needed processing capacity of each VNF, and the capacity of each server are all shown in Fig. 1. For simplification, here we assume the bandwidths of the network connections are large enough and thus do not consider it in this example. We exhibit three possible solutions for this case in Fig. 1, where Fig. 1(a) is the optimal one that minimizes the resource consumption or to say the number of occupied servers. As shown in Fig. 1(a), the capacity of each occupied server, except the last one, is fully utilized. However, there exists back-and-forth data flow between the first two servers, which causes high network latency. Thus, we need to make trade-offs between resource cost and network latency, which is challenging.

Besides, online optimization is also another difficulty because we do not have the future information for the deployment of SFCs. Especially, the unpredictable halfway leaving of SFCs makes the timely deployment hard to close to entirely optimal results over the whole time span.

B. Problem Breakdown: A Two-Stage Scheme

In our model, we put forward a novel concept called *traffic burden*, i.e., the total link latency $l_{i,j}$ of the data flow that needs to be transmitted via the server network. Such a concept is practical and reasonable. For instance, in some advanced datacenter [3], [4], commercial servers can retrieve data of other servers from their shared memory. With such technologies, the data routing problem is eliminated since the network structure here can be considered fully connected. In this case, the network latency is exactly determined by the traffic burden. As for the network with a more complicated topology structure, like the edge network, the total network latency is determined by both the traffic burden generated by the SFCs and the data flow routing in the physical network. Thus, we can break TO-JPR into two stages, using the traffic burden as the pivot. The first stage is to pack VNFs, which can be considered as placing SFCs onto pseudo-servers that are fully connected, with the aim of minimizing the resource cost and the produced traffic burden. We call it the packing stage. The second stage is to map the packed VNF packages or to say the pseudo-servers onto the real servers one by one and route the corresponding transmit data flow between VNF packages, called the mapping-and-routing stage.

Such problem breakdown makes TO-JPR easy to solve. All resource cost relative issues are solved in the packing stage. The mapping-and-routing stage only deals with network latency and can be solved by some modification from the classical solutions of QAP. The conflict between resource cost reduction and network latency minimization in TO-JPR turns into the conflict between resource cost reduction and traffic burden minimization in the packing stage. One of the classical approximation algorithms for BP called the Next-Fit (NF) algorithm [26], gives us enlightenment to solve it.

C. Preliminary Ideas for the Packing Stage

NF is a solution to the problem in the packing stage. The algorithm works as follows. It considers the VNFs in the chaining order. If a VNF fits inside the currently considered server, the VNF is placed on it. Otherwise, the deployment on the current server finishes, a new server is opened and the current VNF is placed on this new server. Fig. 1(b) shows the result of the NF algorithm in the above-mentioned example of TO-JPR. As we all know, the NF algorithm ensures an approximation ratio of 2 for BP. Similarly, it is easy for us to deduce that it can guarantee a small constant ratio for the resource cost objective in TO-JPR as well. Moreover, the NF algorithm puts adjacent VNFs on the same server as much as possible, avoiding the back-and-forth transmit data flow. Thus, the NF algorithm helps reduce traffic burden since it cuts back the number of transmitting data flows. Above all, to some extent, the NF algorithm can make a trade-off between the resource cost and traffic burden. Specifically, it not only assures a small constant approximation ratio for the resource cost objective in TO-JPR but also plays an efficient role in cutting back the traffic burden.

Although NF helps cut back the traffic burden, it is not optimal. We can see the traffic burden of Fig. 1(c) is lower than that of Fig. 1(b), the outcome of NF. What's worse, NF does not have bounds on the produced traffic burden since it cannot directly determine which data flow is idle and which data flow is transmitted. It means some large data flow may need to be transmitted by NF. Thus, we plan to design an improved algorithm for the packing stage, which not only maintains an outstanding bound for the resource cost but also optimally reduces the traffic burden. Only in this way, we can ensure the two stages are not fragmented. That is to say, the performance guarantee of the mapping-and-routing stage can become the global guarantee for the network latency in the whole TO-JPR.

TABLE II
NOTATIONS USED IN THE PACKING STAGE

M_i'	number of used pseudo-servers by DSP for SFC i
$\hat{V}_k^i$	used pseudo-server k for SFC i in DSP
i^j	the sub-SFC of SFC i with VNFs $F_{i,1}, \cdots, F_{i,j}$
P_i^j	the optimal packing scheme of sub-SFC i^j with minimal traffic burden, i.e., $\min \sum_j l_{i,j} \cdot (1 - \sum_{k=1}^{M} x_{i,j}^k \cdot x_{i,j+1}^k)$
T_i^j	the transmitting data flow set of P_i^j
z_i^f	$= 1$, iff the first pseudo-server for new arrived SFC i, i.e., $\hat{V}_1^i$, is merged with the last pseudo-server of another SFC
$Mg^f(i)$	the SFC whose last pseudo-server is merged with the first pseudo-server for new arrived SFC i
z_i^b	$= 1$, iff the last pseudo-server for new arrived SFC i, i.e., $\hat{V}_1^i$, is merged with the first pseudo-server of another SFC
$Mg^b(i)$	the SFC whose first pseudo-server is merged with the last pseudo-server for new arrived SFC i
$\widetilde{T_i^j}$	the transmitting data flow set of OPT for sub-SFC i^j

We extract the core advantages of NF and design an excellent approximation algorithm, called the dynamic segmental packing (DSP) algorithm, for the packing stage. In the following section, we will present DSP in detail and prove that it maintains a small constant approximation ratio of 2 for the resource cost and meanwhile achieves the minimal traffic burden, paving the way for the mapping-and-routing stage.

V. PACKING STAGE

A. Problem Formulation

In this section, we focus on the packing stage. That is, pack VNFs by placing SFCs onto pseudo-servers that are fully connected, with the aim of minimizing the resource cost $\mathbb{R}$ and the produced traffic burden $\mathbb{T}$. Note that $\mathbb{T}$ is the total link latency of all transmitting data flow, i.e.,

$$\mathbb{T} = \sum_{t \in T} \sum_{i=1}^{m} e_i(t) \cdot \left[\sum_{j=1}^{n_i - 1} l_{i,j} \cdot \left(1 - \sum_{k=1}^{M} x_{i,j}^k \cdot x_{i,j+1}^k \right) \right].$$

It can be formulated as below:

$$\begin{aligned} \min \quad & \alpha \mathbb{R} + \beta \mathbb{T} \\ \text{s.t.} \quad & (1) - (2). \end{aligned}$$

Note that the new notations used in this packing stage are shown in Table II.

B. Algorithm Design

Inspired by the Next Fit (NF) strategy, we find putting adjacent VNFs on the same server as much as possible plays a positive role in the resource cost. Thus, we extract this core idea, combined with more designs for data flow, and propose a dynamic segmental packing algorithm, short for DSP, with **time complexity** of $O(N^2)$. Note that a detailed analysis of the time complexity is given in Sec. V-C. The thorough design ideas of DSP are given below and its detailed procedure is shown in Algo. 1.

Instead of taking VNFs as the operation objects like NF, DSP focuses on data flow. Recall that if the adjacent two VNFs are placed on the same pseudo-server, the data flow between them is named as an idle data flow; otherwise, it is called a transmitting data flow. Thus, if all transmitting data flows are determined, we can deduct the packing scheme as follows. Divide each SFC into several segments by the transmitting data flows and pack all VNFs of each segment together, i.e., place all VNFs between two adjacent transmitting data flows on the same pseudo-server. We call such a packing scheme segmental packing. (see Lines 6-11 in Algo. 1)

In order to cut back the resource cost and traffic burden, we need to make the transmitting data flow as less as possible. As for each data flow, we have to determine if it needs to behave as a transmitting data flow or not, under the server capacity constraint.

For each new arrived SFC i, DSP works in real time as below. Note the sub-SFC of SFC i with VNFs $F_{i,1}, \cdots, F_{i,j}$ as sub-SFC i^j, the optimal packing scheme of sub-SFC i^j with minimal traffic burden as P_i^j and the transmitting data flow set of P_i^j as T_i^j. For sub-SFC i^1, it is easy for us to find the optimal packing scheme P_i^1. It is just placing $F_{i,q}$ on a pseudo-server with $l_{i,0}, l_{i,1}$ as transmitting data flow, i.e., $T_i^1 = \{l_{i,0}, l_{i,1}\}$. Denote $T_i^0 = \{l_{i,0}\}$. For each sub-SFC $i^j (1 \le j \le k)$, assume we have recorded transmitting data flow set of the its optimal packing scheme T_i^j. As for sub-SFC i^{k+1}, find the smallest s such that $\sum_{j=s}^{k+1} f_{i,j} \le C$. Then find the $P_i^j (s - 1 \le j \le k)$ with the minimal total traffic burden, noted as P_i^a. Then optimal packing scheme of sub-SFC i^{k+1} is first packing like P_i^a and then packing the remaining VNFs together, i.e., $T_i^{k+1} = \{T_i^a, l_{i,k+1}\}$. (see Lines 1-5 in Algo. 1)

Let us take Fig. 1 for example. Firstly, as for sub-SFC i^1 with only the first VNF $F_{i,1}$, it is easy for us to find the optimal packing scheme P_i^1 with the transmitting data flow set $T_i^1 = \{l_{i,0}, l_{i,1}\} = \{3, 3\}$. As for sub-SFC i^2 with the first two VNFs $F_{i,1}, F_{i,2}$, $f_{i,1} + f_{i,2} = 2 + 1 = 3 < 5$ (the server capacity C), so $s = \min_{\sum_{j=s}^{2} f_{i,j} \le C} s = 1$. As for P_i^0 and P_i^1, $\sum_{l \in T_i^0} l =$

Algorithm 1 Dynamic Segmental Packing (DSP) Algorithm

Input: The VNF size set of new arrived SFC i: $\{f_{i,j}\}$, the link latency set of data flow in SFC i: $\{l_{i,j}\}$ and the unit capacity of servers C.
Output: The packing scheme: $\{server(F_{i,j})\}$, the transmitting data flow set: $T_i^{n_i}$ and the set of used pseudo-servers: $\{\hat{V}_k^i\}_{k=1}^{M_i'}$.

1 $T_i^0 \leftarrow \{l_{i,0}\}, T_i^1 \leftarrow \{l_{i,0}, l_{i,1}\}$;
2 **for** $k = 2 \rightarrow n_i$ **do**
3 Find the smallest s s.t. $\sum_{j=s}^{k} f_{i,j} \le C$;
4 $a \leftarrow \arg\min_{s-1 \le j \le k-1} \sum_{l \in T_i^j} l$;
5 $T_i^k \leftarrow \{T_i^a, l_{i,k}\}$;
6 $k \leftarrow 1, j \leftarrow 1, server(F_{i,j}) = \hat{V}_1^i$;
7 **for** $j < n_i$ **do**
8 **if** $l_{i,j} \in T_i^{n_i}$ **then**
9 $k \leftarrow k + 1$; (If occur transmitting data flow, move to the next pseudo-server)
10 $j \leftarrow j + 1, server(F_{i,j}) = \hat{V}_k^i$;
11 $M_i' \leftarrow k$;
12 Run Inter-Chain Merge (ICM) Algorithm;

$l_{i,0} < l_{i,0} + l_{i,1} = \sum_{l \in T_i^0}$, i.e., P_i^0 has the minimal traffic burden, which is P_i^a. Thus, $a = 0$. In all, as for sub-SFC i^2, the transmitting data flow set of the optimal packing scheme P_i^2 is $T_i^2 = \{T_i^0, l_{i,2}\} = \{l_{i,0}, l_{i,2}\}$. Similarly, as for i^3, $f_{i,1} + f_{i,2} + f_{i,3} = 6 > 5$, $f_{i,2} + f_{i,3} = 4 < 5$, meaning $s = 2$. Then, $a = \arg\min_{1 \le j \le 2} \sum_{l \in T_i^j} l = 1$. So, $T_{i,3} = \{T_i^1, l_{i,3}\} = \{l_{i,0}, l_{i,1}, l_{i,3}\}$. As for i^4, $s = 4$, $a = \arg\min_{3 \le j \le 3} \sum_{l \in T_i^j} l = 3$, $T_{i,4} = \{T_i^3, l_{i,4}\} = \{l_{i,0}, l_{i,1}, l_{i,3}, l_{i,4}\}$. As for i^5, $s = 5$, $a = \arg\min_{4 \le j \le 4} \sum_{l \in T_i^j} l = 4$, $T_{i,5} = \{T_i^4, l_{i,5}\} = \{l_{i,0}, l_{i,1}, l_{i,3}, l_{i,4}, l_{i,5}\}$, which is just the traffic burden set of the optimal packing scheme of the given SFC *i*, as shown in Fig. 1 (c).

Above we have finished the discussion of in-chain resource management. It ensures any two adjacent VNF packages (or to say, pseudo-servers) among a SFC cannot be merged together under server capacity limitation. Next, we design an inter-chain merge (ICM) algorithm to further reduce the resource cost by inter-chain resource management. The main target of the algorithm design is to ensure two "adjacent" VNF packages between SFCs cannot be merged together under server capacity limitation. If the order of SFCs is provided, this goal can be achieved through the conditional merge procedure introduced below. Thus, the remaining challenge is determining the order of the SFCs. A natural approach is to follow their arrival order. However, this approach may lead to inefficiencies, as previously occupied but now idle servers might be far from recently occupied servers, making them difficult to reuse due to high latency. To address this issue, prioritize newly arrived SFCs to take the place of the timeout SFCs and otherwise order them by their arrival time. In particular, we use a set, U, to record the SFCs that have left and use a doubly-linked list, O to represent the timely order of SFCs. Each time a new SFC i' leaves, if it is the first or last node in list O, we remove it from the list; otherwise, we put it into the set U. Each time a new SFC i arrives, if U is empty, we just add this SFC i after the last node of O; otherwise, we randomly move an SFC o out of U and use the new SFC i to replace the location of SFC o in O. Then we will do a conditional merge depending on the new order of SFCs in O. Specifically, if this new SFC i has the previous node s in O and SFC $s \notin U$, check whether the first VNF package of SFC i and the last VNF package of its previous SFC s can be merged together under server capacity limitation. If so, merge them and record such forward merge by $z_i^f = 1$ and $Mg^f(i) = s$. Similarly, if this new SFC i has the next node o in O and SFC $o \notin U$, check whether the last VNF package of SFC i and the first VNF package of its next SFC o can be merged together. If so, merge them and record such backward merge by $z_i^b = 1$ and $Mg^b(i) = o$. Note, initially $z_i^f = z_i^b = 0$. The detailed procedure of ICM is shown in the appendix.

C. Complexity Analysis

DSP has the **time complexity** of $O(N^2)$. We will explain in detail below.

In Algo. 1, Line 3 checks the number of chaining VNFs connected with VNF $F_{i,j}$ which can be contained in a server while Line 4 finds a sub-SFC which has the minimal traffic burden among the sub-SFCs before these VNFs. So, the computational complexity of Line 3 and Line 4 are both the number of those VNFs which can be contained in a server, which is no more than the number of VNFs in SFC *i*, n_i, i.e.,

$$\text{timeComplexity(Line 3)} = \text{timeComplexity(Line 4)} = O(n_i).$$

Thus, the time complexity of the cycle of Line 2-5 is $O(n_i^2)$.

Lines 7-11 place the VNFs in SFC *i* onto the pseudo-servers one by one according to the transmitting data flow set T_i^i obtained before. Thus, the time complexity here is $O(n_i)$.

Line 12 runs the ICM algorithm with the time complexity $O(1)$.

In all, the time complexity of DSP for SFC i is $O(n_i^2 + n_i + 1) = O(n_i^2)$ and for all SFCs is $O(n_1^2 + \cdots + n_m^2) \le O(N^2)$.

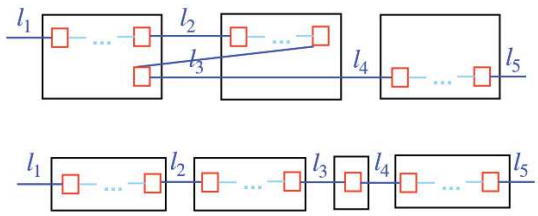


Fig. 2. An example of converting one OPT to the segmental packing version.

D. Proof of Minimal Traffic Burden

We now prove DSP is an optimal solution (OPT) for traffic burden, having minimal traffic burden, in Theorem 2. Note that the OPT we mention in this section is the optimal packing solution with minimal traffic burden, which is different from the OPT for resource cost in Section V-E.

Theorem 2: DSP is an OPT for traffic burden.

Before proving Theorem 2, we first prove the below Lemma.

Lemma 3: There must exist an OPT of traffic burden, which is produced by segmental packing.

Proof: If some OPT of traffic burden is not produced by segmental packing, we can repack it by segmental packing based on the transmitting data flow set. Specifically, all VNFs between two transmitting data flows are placed on a new pseudo-server. Fig. 2 gives an example. Then the new repacking solution is also an OPT of traffic burden since the transmitting data flow set does not change. □

Based on Lemma 3, we only consider the OPT produced by segmental packing, in the below proof of Theorem 2.

Proof: To prove Theorem 2, we only need to prove $T_i^{n_i}$ is the transmitting data flow set of OPT for SFC *i*. Note that, here we only consider the time slot *t* when SFC *i* exists in the system, i.e., $e_i(t) = 1$.

We will prove by mathematical induction to show for each *k*, T_i^k is the transmitting data flow set of OPT for sub-SFC i^k. It is easy to check the base case. Then we only need to prove the inductive step that T_i^{k+1} is the transmitting data flow set of OPT for sub-SFC i^{k+1}, noted as $\widehat{T_i^{k+1}}$. Suppose in the transmitting data flow set of OPT for sub-SFC i^{k+1}, the previous one before $l_{i,j}$ is $l_{i,x}$, as shown in Fig. 3.

- *Claim 1:* $\widetilde{T_i^{k+1}} = \{T_i^x, l_{i,k+1}\}$.
- *Claim 2:* $x = a = \arg\min_{l-1 \le j \le k} \sum_{d \in T_i^j} d$.

(Proof omitted here, please refer to the appendix.)

Combined with Claim 1 and Claim 2, we finally prove that

$$\widetilde{T_i^{k+1}} = \{T_i^x, l_{i,k+1}\} = \{T_i^a, l_{i,k+1}\} = T_i^{k+1}.$$

□

E. Proof of Approximation Ratio for Resource Cost

We now demonstrate DSP achieves a small constant approximation ratio to the optimal solution (OPT) on the resource cost $\mathbb{R}$, i.e., $C \times M'$ by the following two theorems. Note that the proofs are omitted here, please refer to the appendix.

Theorem 4: $\mathbb{R}_{DSP} < \left(2 + \frac{1}{\gamma}\right) \cdot \mathbb{R}_{OPT}$, *where* $\gamma = min_{1 \le i \le m} \frac{\sum_{j=1}^{n_i} f_{i,j}}{C}$.

Theorem 5: In the special online cases, when SFCs queue to leave in their arriving order,

$$\mathbb{R}_{DSP} < 2 \cdot \mathbb{R}_{OPT} + C \cdot T_0,$$

where $T_0 = \{t \in T | \sum_{i=1}^{m} e_i(t) > 0\}$.

VI. Mapping-and-Routing Stage

A. Problem Formulation

In this section, we will deal with the mapping-and-routing stage. Our task is to map the packed VNF packages onto the real servers one by one and route the corresponding transmitting data flow between the VNF packages, with the goal of minimizing network latency.

To formulate the problem in this stage, we first redefine two *Boolean* variable $\hat{x}_{i,k}^p$ and $\hat{w}_{i,k}^{p,q}$, which are the decision variables in this problem. Note that $\hat{x}_{i,k}^p = 1$ if and only if pseudo-server $\hat{V}_k^i$ is placed on server V_p; $\hat{w}_{i,k}^{p,q} = 1$ if and only if the output data flow of pseudo-server $\hat{V}_k^i$ passes through network connection (V_p, V_q).

Besides, denote the link latency of the output data flow of $\hat{V}_k^i$ as $\hat{l}_{i,k}$. Then we can formulate the mapping-and-routing problem as below.

$$\min \mathbb{L} = \sum_{t \in T} \sum_{i=1}^{m} \sum_{k=1}^{M_i'-1} \sum_{p,q=1}^{M+r} e_i(t) \cdot \hat{l}_{i,k} \cdot \hat{w}_{i,k}^{p,q}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \sum_{k=1}^{M_i'} \hat{x}_{i,k}^p \le 1 + \sum_{i=1}^{m} (z_i^f x_{i,1}^p + z_i^b x_{i,M_i'}^p), \quad \forall p,$$

$$\sum_{p=1}^{M} \hat{x}_{i,k}^p = 1, \quad \forall 1 \le i \le m, 1 \le k \le M_i',$$

$$\sum_{p=1}^{M} \hat{w}_k^{p,s} - \sum_{q=1}^{M+r} \hat{w}_k^{s,q} = \hat{x}_{k+1}^s - \hat{x}_k^s, \quad \forall k, s,$$

$$\sum_{k=1}^{M'-1} (\hat{w}_k^{p,q} + \hat{w}_k^{q,p}) \cdot \hat{l}_k \le B_{p,q}, \quad \forall\ 1 \le p < q \le M,$$

TABLE III
Notations Used in the Mapping-and-Routing Stage

$\hat{x}_{i,k}^p$	$= 1$, iff pseudo-server $\hat{V}_k^i$ is placed on server V_k
$\hat{w}_{i,k}^{p,q}$	$= 1$, iff the output data flow of pseudo-server $\hat{V}_k^i$ pass through network connection (V_p, V_q)
$\hat{l}_{i,k}$	link latency of the output data flow of pseudo-server $\hat{V}_k^i$
$\hat{d}_{i,k}$	the number of hops that the output data flow of $\hat{V}_k^i$ passes through after employing proposed algo., i.e., $\sum_{p,q=1}^{M+r} \hat{w}_{i,k}^{p,q}$
$\hat{l}_{i,k}^*$	link latency of data flow between pseudo-servers $\hat{V}_k^i$ and $\hat{V}_{k+1}^i$ after employing OPT
$\hat{d}_{i,k}^*$	the number of hops that the output data flow of $\hat{V}_k^i$ passes through after employing OPT

$$(x_{i,1}^p - x_{Mg^f(i),M'_{Mg^f(i)}}^p) \cdot z_i^f = 0, \quad \forall i, p,$$

$$(x_{i,M_i'}^p - x_{Mg^b(i),1}^p) \cdot z_i^b = 0, \quad \forall i, p$$

where the first four constraints correspond to Eq. 1–4, while the last two constraints imply the merge requirements from the results of ICM.

Note that the new notations used in this mapping-and-routing stage are shown in Table III.

B. Ratio of Network Latency Based on DSP

In this section, we first analyze how DSP helps cut back network latency after minimizing the total traffic burden. Based on the property of DSP (Thm. 2), it is easy to reach the following theorem.

We denote the number of hops that the output data flow of $\hat{V}_k^i$ passes through (i.e., $\sum_{p,q=1}^{M+r} \hat{w}_{i,k}^{p,q}$) after employing DSP+(any feasible mapping-and-routing algorithm) and OPT as $\hat{d}_{i,k}$ and $\hat{d}_{i,k}^*$, respectively. Additionally, we denote the link latency of data flow between pseudo-servers $\hat{V}_k^i$ and $\hat{V}_{k+1}^i$ after employing DSP+(any feasible mapping-and-routing algorithm) and OPT as $\hat{l}_{i,k}$ and $\hat{l}_{i,k}^*$, respectively.

Theorem 6: Based on DSP, any feasible mapping-and-routing algorithm can maintain an approximation ratio of d_G *on the network latency, i.e.*,

$$\mathbb{L}_{DSP+} \le \max \hat{d}_{i,k} \cdot \mathbb{L}_{OPT} \le d_G \cdot \mathbb{L}_{OPT},$$

where $\hat{d}_{i,k}$ *is the number of hops that the output data flow of* $\hat{V}_k^i$ *passes through (i.e.,* $\sum_{p,q=1}^{M+r} \hat{w}_{i,k}^{p,q}$*) after employing the feasible mapping-and-routing algorithm and* d_G *is the diameter of the network graph* G.

Proof: The proof can be found in the appendix. □

By this theorem, we can get the conclusion that in general, DSP maintains the approximation ratio of d_G on network latency. It is worth noticing that the diameter d_G is a parameter depending on the topology structure and scale of the network graph. Work [29] shows the diameters of sparse random graphs grow logarithmically with the network scale. Specifically, it forms the expression of $c \cdot lnM + o(lnM)$, where c is the constant depending on the expected degree and M is the number of nodes in the graph. Besides, as for some specific types of network topologies, such as ring, fat tree, star, and mesh, the network diameter has a relation with the network scale, the number of network nodes M, as follows. The diameter

of the ring, fat tree, star, and mesh network is respectively $\frac{M}{2}$, $\log_2(M)$, 2, 1. In the worst case, the graph is chained with a diameter of *M*. But in practice, the network diameter is also an important parameter for the network, and network providers usually pursue the lowest possible network diameter when deploying servers and connections. The chained or ring network usually is not employed in practice. Thus, d_G is typically $O(logM)$.

Moreover, if the mapping-and-routing algorithm can further bound the maximal number of hops of any transmitting data flow passing through, our two-stage scheme can obtain a tight performance bound on network latency. Below, we first propose a heuristic algorithm called greedy mapping (GM) algorithm to reduce the number of hops of all transmitting data flow as much as possible. It can cooperate with DSP in an online manner, for solving the mapping-and-routing sub-problem.

C. A Heuristic Algorithm for Mapping-and-Routing Stage: Greedy Mapping (GM) Algorithm

Inspired by the nearest neighbor (NN) algorithm, the most popular heuristic algorithm for QAP, we propose a greedy mapping (GM) algorithm. Specifically, after the inter-chain merge (ICM) in DSP, there are four cases for each new SFC *i*, based on if the forward merge and backward merge happen, i.e., if $Mg^f(i) > 0$ or not and if $Mg^b(i) > 0$ or not. For example, if $Mg^f(i) > 0$, the first VNF package of SFC *i* must be placed on the server where the last VNF package of its previous SFC $Mg^f(i)$ is placed. In this case, the mapping server of the first VNF package of this new SFC has been determined. Similarly, if $Mg^b(i) > 0$, the mapping server of the last VNF package of SFC *i* is also determined.

As for the case that the mapping nodes of both the first and last VNF package of SFC *i* are both determined, we aim to find a short path from V_p to V_q passing through at least $M_i' - 2$ idle server nodes on *G*. Specifically, we first find the shortest path between V_p and V_q, noted as *P*. If there are at least $M_i' - 2$ idle servers on *P*, *P* is the satisfying path. If not, continue adding the nearest idle server node outside *P* to *P*, i.e., inserting it between its two nearest nodes on *P*, until there are enough idle servers on *P*. Finally, we map $\hat{V}_1^i$ to V_p, map $\hat{V}_{M_i'}^i$ to V_q and map $\hat{V}_2^i, \cdots, \hat{V}_{M_i'-1}$ to the $M_i' - 2$ idle servers on path *P*;

As for the other three cases, we determine the mapping of other VNF packages based on the nearest neighbor algorithm. Specifically, if the mapping server of either the first or the last VNF package is determined, use this server as the starting node. If neither of them is determined, just randomly choose an idle server as the starting node. Then following the VNF chaining order, sequentially find the nearest connected idle server, under bandwidth limitation, as the mapping node for the next VNF package. Note if some network connection has not enough rest bandwidth for the data flow, we consider this network connection as disconnected when routing this data flow.

In sum, the detailed procedure of GM with a time complexity of $O(M^3)$ is shown in Algo. 2.

Algorithm 2 Greedy Mapping (GM) Algorithm (employed After DSP for Each SFC i)

Input: The new arrived SFC i, the physical network G, the packing scheme: $\{server(F_{i,j})\}$, z_i^f and z_i^b, $Mg^f(i)$ and $Mg^b(i)$.
Output: The placement scheme $server(F_{i,j})$.

1 **if** $z_i^b = 0$ **then**
2 **if** $z_i^f = 0$ **then**
3 Randomly find an idle server node, noted as $currentServer$.
4 **else**
5 $s \leftarrow Mg^f(i)$, $currentServer \leftarrow server(F_{s,n_s})$;
6 Map $\hat{V}_1^i$ to $currentServer$ and renew $server(F_{i,\cdot})$;
7 **for** $k = 2 \rightarrow M_i'$ **do**
8 Under bandwidth limitation, find the nearest connected idle server node from $currentServer$, noted as V_p;
9 Map $\hat{V}_k^i$ to V_p and renew $server(F_{i,\cdot})$;
10 $currentServer = V_p$;
11 **else**
12 **if** $z_i^f = 0$ **then**
13 $s \leftarrow Mg^b(i)$, $currentServer \leftarrow Server(F_{s,1})$;
14 Map $\hat{V}_{M_i'}^i = \hat{V}_1^s$ to $currentServer$ and renew $server(F_{i,\cdot})$;
15 **for** $k = M_i' - 1 \rightarrow 1$ **do**
16 Under bandwidth limitation, find the nearest connected idle server node from $currentServer$, noted as V_p;
17 Map $\hat{V}_k^i$ to V_p and renew $server(F_{i,\cdot})$;
18 $currentServer = V_p$;
19 **else**
20 $s \leftarrow Mg^f(i)$, $V_p \leftarrow Server(F_{s,n_s})$;
21 $o \leftarrow Mg^b(i)$, $V_q \leftarrow Server(F_{o,1})$;
22 Find the shortest path P between V_p and V_q;
23 **while** *num. of idle servers on* $P < M_i' - 2$ **do**
24 Under bandwidth limitation, find the nearest idle server outsider P and add it to P by inserting between its two nearest nodes on P;
25 Map $\hat{V}_1^i$ to V_p and $\hat{V}_{M_i'}^i$ to V_q;
26 Map $\hat{V}_2^i, \cdots, \hat{V}_{M_i'-1}$ to the $M_i' - 2$ idle servers on path P;
27 Renew $server(F_{i,\cdot})$, correspondingly.

VII. Extension for More Generalized Scenarios

A. Scenarios With Heterogeneous Servers

In the above context, we have focused on data center scenarios and assumed that all the servers have uniform processing capacity, denoted as *C*. In this section, we extended our model and proposed algorithms for heterogeneous networks, where the processing capacity of server V_k is denoted as C_k.

For SFC deployment in such scenarios, we must revise the recursive steps of DSP because the key step (Line 3 in Algo. 1) does NOT stand under diverse server capacities. We

propose a revised algorithm for the packing stage, referred to as r-DSP, which builds upon the core principles of the previously proposed DSP algorithm. Additionally, our two-stage solution framework undergoes slight modifications. In particular, we first generate a Greedy Routing path using the nearest-neighbor strategy and then apply r-DSP to place VNFs, integrating both packing and mapping, rather than following the previous two-stage approach of first packing and then performing mapping and routing in uniform scenarios. The main idea of our new algorithm design is as follows:

For each newly arrived SFC i, if all VNFs can be successfully placed on a single idle server, they are allocated accordingly. Otherwise, a Greedy Routing (GR) algorithm is first applied to generate a routing path. In detail, if the first VNF $F_{i,1}$ of SFC i can be successfully on the server V_k, where the last VNF of the previous SFC $i-1$ ($F_{i-1,n_{i-1}}$) was placed, the routing path P starts from this server V_k; otherwise, the routing path P starts from a randomly selected idle server. Next, the algorithm iteratively selects the nearest connected idle server node from the current server to extend the routing path P until no idle server remains in the network. The servers along the routing path P are sequentially denoted as $V_{P_1}, \cdots, V_{P_{|P|}}$. The size of server V_{P_k} is represented as C_{P_k} while the number of hops between $V_{P_{k-1}}$ and V_{P_k} is denoted as d_{P_k}, with the initial hop count set as $d_{P_1} = 0$.

Algorithm 3 Revised-DSP (r-DSP) Algorithm

Input: The new arrived SFC i with parameters $\{f_{i,j}\}$ and $\{l_{i,j}\}$, and the routing path P with parameters $\{C_{P_k}\}$ and $\{d_{P_k}\}$.
Output: The placement scheme: $\{server(F_{i,j})\}$.

```
T_i^0 ← {l_{i,0}}, T_i^1 ← {l_{i,0}, l_{i,1}};
for k = 2 → n_i do
    a ← k − 1;
    for s = 1 → k do
        if Σ_{j=s}^{k} f_{i,j} ≤ C_{P_{|T_i^{s-1}|}} (condition 1) and
        Σ_{T_i^{s-1}} < Σ_{T_i^a} (condition 2) then
            a ← s − 1
    T_i^k ← {T_i^a, l_{i,k}};
k ← 1, j ← 1, server(F_{i,j}) = V̂_1^i = V_{P_1};
for j < n_i do
    if l_{i,j} ∈ T_i^{n_i} then
        k ← k + 1;
    j ← j + 1, server(F_{i,j}) = V̂_k^i = V_{P_k};
M_i' ← k;
```

The design ideas of r-DSP are given below and its detailed procedure is shown in Algo. 3. r-DSP follows the recursive framework of DSP. Specifically, the optimal packing scheme of sub-SFC i^k must be generated from the optimal packing scheme of sub-SFC $i^{s-1}(1 \le s \le k)$, combined with a new VNF package containing all remaining VNFs $F_{i,s}, \cdots, F_{i,k}$. The key difference lies in the feasibility check for packing and mapping VNFs $F_{i,s}, \cdots, F_{i,k}$ onto a new idle server. Without a uniform server size, the decision depends on the specific server capacity $C_{P_{|T_i^{s-1}|}}$ on the routing path P rather than the uniform server size. Consequently, we modify the recursive steps of DSP (Lines 3-4 in Algorithm 1) and introduce new recursive steps based on diverse server capacities, as demonstrated in Lines 3-6 of Algorithm 3. Here, conditions 1 and 2 correspond to Lines 3 and 4 in Algorithm 1, respectively.

Claim 7: Following the same proof approach, Theorems 4 and 5 still hold for the r-DSP algorithm, ensuring a small constant approximation ratio for resource cost.

Theorem 8: The r-DSP algorithm, combined with any feasible routing algorithm, can also maintain an approximation ratio of $O(d_G)$ on the network latency, i.e.,

$$\mathbb{L}_{r-DSP+} \le O(d_G) \cdot \mathbb{L}_{OPT}.$$

where d_G is the diameter of the network graph G.

Proof: The proof can be found in the appendix. □

B. Scenarios With Shareable VNFs

In the context above, we have focused on non-sharing scenarios where VNF instances cannot be shared among different services, protecting data security and privacy. However, many application scenarios allow VNF instances to be shared across different services, reducing resource costs associated with separate management and maintenance. To address this, we extend our model by introducing a new integer variable $h_{i,j} \in H$ to represent the types of shareable VNFs, where H is the set of shareable VNF types. For instance, if $H = \{1, 2, 3\}$, it indicates there are three types of shareable VNFs and $h_{i,j} = 3$ means VNF $F_{i,j}$ is a type-3 shareable VNF. For those VNFs $F_{i,j}$ that are not allowed to be shared among different services, we set $h_{i,j} = 0$. For each type $h \in H$ of shareable VNFs, if it is shared between two services, f'_h computing resources will be saved. In other words, when placing two shareable VNFs of the same type on the same server, the total computing resources required will be f'_h less than the original.

We then propose a heuristic Sharing-First algorithm based on our previous GR+r-DSP algorithm, denoted as sf-DSP, to deploy SFCs under the extended model with shareable VNFs. It works as follows. Suppose there are s types of shareable VNFs, like Firewall, Load Balancer (LB), Deep Packet Inspection (DPI), etc. Denoted the set of all newly arrived SFCs at time slot t as I_t. For each shareable VNF $F_{i,j}(i \in I_t)$ among these newly arrived SFCs, check if there is an active server V_k with the same type of shareable VNFs placed on it and if this shareable VNF $F_{i,j}$ can be successfully placed on such active server V_k. If so, place VNF $F_{i,j}$ on this server V_k and mark shareable VNF red. After checking all new shareable VNFs, mark all unmarked shareable VNFs blue.

Following the chaining order of each SFC and the index order of newly arrived SFCs, we obtain the sub-SFCs and deploy them one by one using the GR+r-DSP approach. Each sub-SFC starts from the first VNF of each SFC or a red shareable VNF, and ends with the first VNF after the starting VNF, satisfying one of the following three conditions: (1) the next VNF is a red shareable VNF; (2) the next VNF is a blue shareable VNF, and there are more than one blue shareable VNFs of the same type among the newly arrived SFCs; (3) it

is the last VNF of the SFC. If the ending condition is (2), after deploying this sub-SFC with GR+r-DSP, if all blue shareable VNFs of the same type as the next VNF can be successfully placed on the current server, they are allocated accordingly. Otherwise, we find the nearest largest idle server from the current server to accommodate the next VNF and as many blue shareable VNFs of the same type as possible, while adhering to the server capacity constraints. If placement is successful, the blue shareable VNF is re-marked as red. This process is repeated until all VNFs of the newly arrived SFCs are placed.

To further enhance resource management, when deploying sub-SFC i using the GR+r-DSP approach, we relocate the last VNF package $\hat{V}^i_{M'_i}$ from the idle server $V_{P_{M'_i}}$ to an active server V_k if such an active server V_k exists: (1) it has sufficient remaining computing resources for the VNF package $\hat{V}^i_{M'_i}$, and (2) it is no farther from the previous server $V_{P_{M'_i-1}}$ than the idle server $V_{P_{M'_i}}$. This procedural step serves a similar function to the ICM algorithm.

VIII. Performance Evaluation

A. Simulation Setup

The evaluation of our proposed algorithms in different scenarios is performed through simulations. In simulations, we adopt a Monte Carlo simulator by adding our model to the simulator framework, because it is a good simulator suited for the variability of the targeted environment according to [30]. As for the infrastructure of the model, we set the bandwidth capacity of each hop in the edge network, $B_{p,q}$, as 1300 Mbps (the bandwidth of Wireless 802.11ac) if $(V_p, V_q) \in E$. The flow rate of SFC i after passing VNF $F_{i,j}$, $b_{i,j}$, ranges from 0.5 Mbps to 5 Mbps. And we use the Cisco EIGRP link weight function [25] to calculate $l_{i,j}$ by setting only K_2 to a nonzero constant.[2] Additionally, we implement the online mechanism by Poisson distribution. In detail, we set $t^a_i \sim possion(3)$, $t_i \sim \min\{\mathbb{U}(1,10), 10 - t^a_i\}$.

Note that for each group of outcomes, we use the average value from 100 groups of simulations to avoid the impact of extreme cases. And the errors shown in all the plots below are determined by the standard variances of the corresponding 100 groups of simulations.

B. Practical Approximation Ratios

In Theorem 4, 5, 6, we proved the worst-case approximation ratios of our algorithms. In this section, we perform extensive simulations to compare the performance of our algorithms with the optimal solutions (OPTs) and show some average-case approximation ratios.

1) Network Topology and SFC Setting: In order to make finding OPTs possible, we first consider a relatively small-scale TO-JPR with only 4 SFCs, 20 VNFs, and 15 servers. Specifically, we set $m = 4, n_1 = \cdots = n_4 = 5, M = 15, C = 4, f_{i,j} \sim \mathbb{N}(1, 0.25)$.

Note that even on such a small scale, it is impossible to achieve accurate OPTs. Because if accurately computing OPT by enumerating, $15^{20} \approx 3.3 \times 10^{23}$ instances need to be computed. Even if each enumeration required one CPU clock, it would cost a 2-GHz computer nearly 5×10^6 years to complete a task. Thus, we use *MIP solver* to approximate the results of OPTs in the simulations.

[2]In order to make the summed resource cost and network latency in the same order of magnitude, we set $K_2 = 1/1300$.

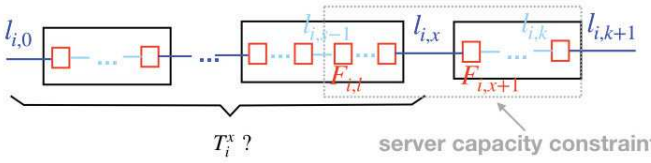


Fig. 3. Diagram for proving Theorem 2.

In our simulations, we will try 5 different groups of weight parameter (α, β) on a mesh topology to evaluate the performance of our algorithms, comparing with different OPTs. Specifically, we pick 5 representative OPTs as follows: **OPT1**: $\alpha = 1, \beta = 10$; **OPT2**: $\alpha = 1, \beta = 2$; **OPT3**: $\alpha = 1, \beta = 1$; **OPT4**: $\alpha = 2, \beta = 1$; **OPT5**: $\alpha = 10, \beta = 1$. Additionally, we produce 5 classical topology structures of the ring, fat tree, hybrid, star, and mesh, for simulations (as shown in Fig. 7).

Here we use box plots to show our results. And the approximation ratios in the plot in this section (Fig. 4, 5) are computed by dividing the corresponding cost of our proposed algorithms by that of those OPTs in each simulation.

2) Performance Comparison With OPTs: Fig. 4 gives the performance ratios of DSP+GM based on the different OPTs with different weight parameters on the mesh network. In the first plot, we can see the average ratios of the total cost by our algorithms to that of different OPTs are all smaller than 1.25, which reveals the superiority of our algorithm. In the second plot of Fig. 4, the average ratios of resource cost are between $1.15 - 1.25$, all much low than $2 + \frac{1}{\gamma}$, the theoretical resource approximation ratio proved in Thm. 4. In the third plot, we can see the ratios of traffic burden are all less than 1, consistent with the conclusion in Thm. 2 that DSP produces the minimal traffic burden. In the fourth plot, the ratios of latency here are the same as that of traffic burden in the third plot. This is because the diameter of the mesh network is 1. By Thm. 6, the obtained latency by DSP+GM is just the optimal one.

Fig. 5 gives the performance ratios of our algorithms to the OPT with $\alpha = \beta = 1$, on 5 different classical network topology structures. In the first plot, we can see the average ratios of the total cost are all smaller than 1.25, which reveals our algorithm can preserve its superiority based on different network topologies. The same as Fig. 4, the second and third plots of Fig. 5 respectively verify 4 and Thm. 2. In the fourth plot, the average ratios of latency on different topologies are all smaller than 1.35, far lower than the corresponding network diameters, the theoretical bound proven in Thm. 6. This is because the theoretical approximation ratio of network latency in Thm. 6 is a worst-case upper bound. In practice, the real latency ratios are typically much smaller than the network diameter.

C. Simulations on 4 Large-Scale Real Network Topologies

1) Network Topology and Simulation Setup: In this section, we perform simulations on 4 large-scale real network

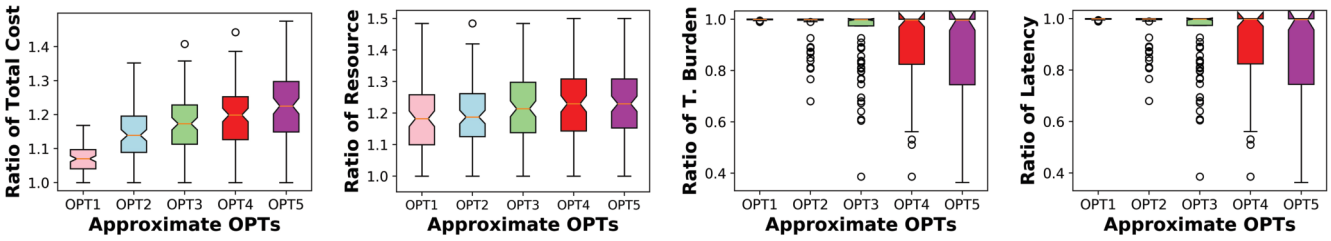


Fig. 4. Different approximation ratios based on different OPTs with different weight parameters (OPT1: $\alpha = 1, \beta = 10$, OPT2: $\alpha = 1, \beta = 2$, OPT3: $\alpha = 1, \beta = 1$, OPT4: $\alpha = 2, \beta = 1$, OPT5: $\alpha = 10, \beta = 1$), running on Mesh topology.

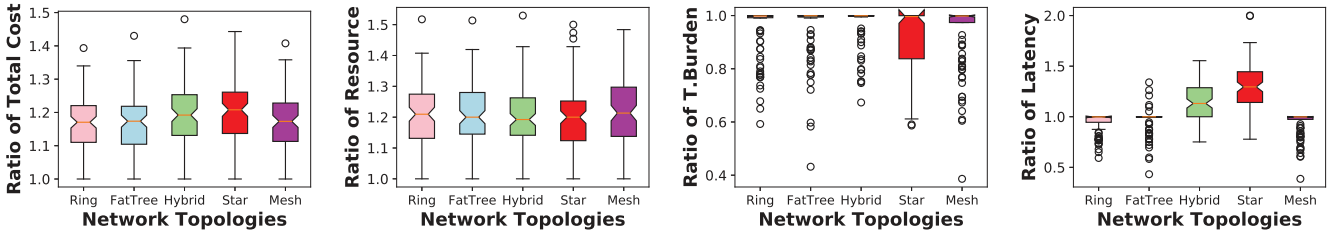


Fig. 5. Different approximation ratios on different network topologies (Set $\alpha = \beta = 1$).

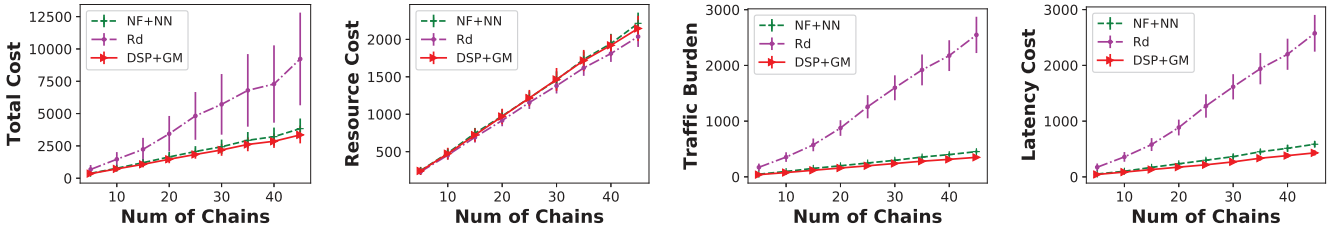


Fig. 6. Performance comparisons with benchmarks.

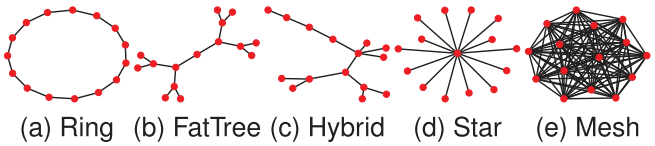


Fig. 7. 5 different Network topologies with 15 Nodes.

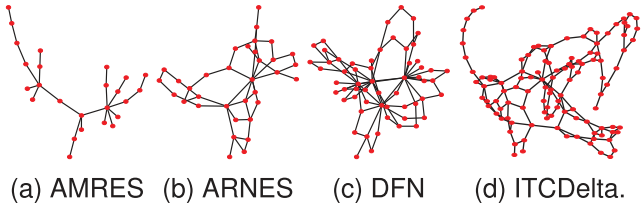


Fig. 8. Different real network topologies.

topologies, shown in Fig. 8, 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). We set $\alpha = \beta = 1, m \in \{5, 10, \cdots, 45\}, n_i = 8, M = 113, C = 4, f_{i,j} \sim \mathbb{N}(1, 0.25)$.

We will evaluate the performance of DSP by comparing it with the two benchmarks:(1) **Rounding Algorithm (Rd)**,[3] (2) **Next Fit and Nearest Neighbour (NF+NN) algorithm**.[4]

2) Performance Comparisons With Benchmarks: Fig. 6 show the performance of our designed algorithms compared with two benchmarks on the network topology called ITCDeltacom. Firstly, we can see DSP+GM always achieves a better performance than the benchmarks in terms of the total cost. Then, when we take a deep sight at their respective performance on resource cost and the network latency, it is interesting to find the fact that Rd is the one with the lowest resource cost but performs worst on the network latency and the total cost. Conversely, our proposed algorithms achieve the smallest network latency and a reasonable resource cost, slightly worse than that of the Rd algorithm, thereby achieving the lowest total cost. It reveals that if optimizing to the limit, the two optimization objectives are conflicting. And minimizing the resource cost is not a good choice for TO-JPR. Besides, the similarity of the third and fourth plots shows the

[3]After formulating TO-JPR as an ILP, the rounding algorithm can be used here. It is a classical approach to obtaining a provable bound for ILP. See [9], [10] and [24] as an example. Note it is a batched algorithm.

[4]NF and NN are the two most popularly-used greedy strategies in the packing and routing problems respectively. See work [22], [23] as an example. It is an online algorithm.

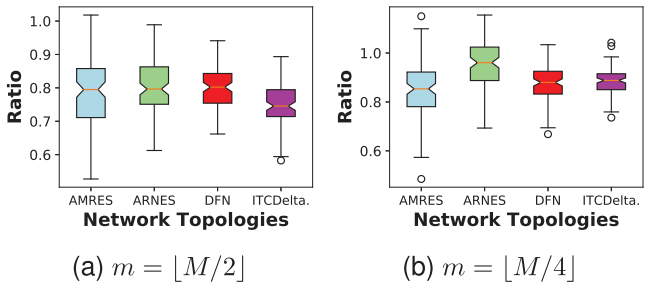


Fig. 9. Relative performance comparisons between DSP+GM and NF+NN on 4 different real network topologies.

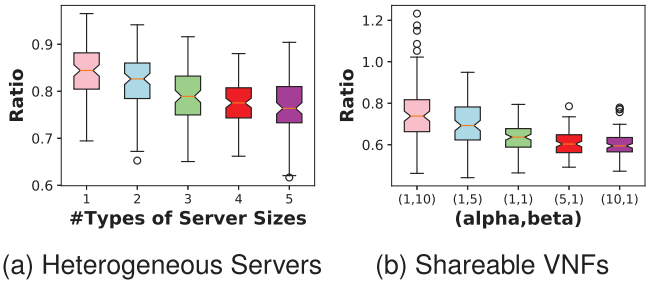


Fig. 10. Performance evaluation in generalized scenarios.

tight relationship between the traffic and the network latency. Combined with the above interesting fact, it demonstrates minimizing traffic burden is a good strategy to balance the resource cost and network latency, which can ensure a pretty low latency and a reasonable resource cost and thereby achieving a low total cost.

Since NF+NN is a better benchmark, we further analyze the competitive performance ratio of DSP+GM to NF+NN in Fig. 9. From this figure, we can see that in scenarios with more SFCs, DSP+GM outperforms NF+NN by an average of 22% and up to 51% (in the case of the AMRES topology). In scenarios with fewer SFCs, DSP+GM outperforms NF+NN by an average of 11% and up to 47% (also in the case of the AMRES topology). In general, DSP+GM still makes significant progress compared with NF+NN. The more SFCs there are, the greater the improvement. In some special worst cases, DSP+GM shows a significant performance improvement over NF+NN, nearing 50%. This is because DSP+GM has a better performance guarantee in the worst cases.

D. Simulations for More Generalized Scenarios

1) Network Topology and Simulation Setup: In this section, we perform simulations on a large-scale real network topologies named DFN with 1 to 5 types of server sizes, i.e., $Type \in [[4],[3.5,4.5],[3,4,5],[2.5,3.5,4.5,5.5],[2,3,4,5,6]]$. The size of each server C_k is randomly selected from the corresponding *Type*. Besides, we set $\alpha = \beta = 1, m = 24, n_i = 8, M = 58, f_{i,j} \sim \mathbb{N}(1, 0.25)$. In our simulations for scenarios with shareable VNFs, we additionally set $Type = [2,3,4,5,6], H = \{1,2,3\}$, $f'_1 = 0.1, f'_2 = 0.3, f'_3 = 0.5$ and each VNF has 10% chance of being shareable, with a randomly selected shareable type from H. Since NF+NN serves as a better benchmark for DSP+GM, we use it as the baseline and calculate the competitive performance ratio of the extended algorithms to NF+NN for performance evaluation, as shown in Fig. 10.

2) Performance in Scenarios With Heterogeneous Servers: Fig. 10(a) shows in scenarios with heterogeneous servers, our extended algorithm (GR+r-DSP) outperforms NF+NN by an average of 16-23 % across different heterogeneous settings. Overall, the extended algorithm demonstrates significant improvements compared to NF+NN, exhibiting better performance in scenarios with a larger number of server size types, thus highlighting its strong adaptability to diverse server environments.

3) Performance in Scenarios With Shareable VNFs: Fig. 10(b) illustrates the effectiveness of our other extended algorithm, sf-DSP, in scenarios with shareable VNFs under different weight parameter settings. As shown in Fig. 10(b), we can see sf-DSP achieves an improvement of 19-25% over the baseline NF+NN. The larger the ratio $\frac{\alpha}{\beta}$, the better improvement. This is because the sharing-first rule of sf-DSP enhances performance by effectively reducing resource costs.

IX. Conclusion and Future Work

We put forward an online traffic-sensitive joint SFC placement and flow routing model, optimizing both resource cost and network latency. To address this issue, we propose a two-stage scheme based on a novel and practical concept of traffic burden. In the first stage, we propose DSP to pack VNFs. Importantly, DSP not only maintains a minimal traffic burden but also achieves a small approximation ratio on the resource cost. In the second stage, our task is to map the packed VNF packages onto the real servers one by one and implement data flow routing between them. As for this stage, we design a greedy mapping (GM) algorithm for the general online case, which obtains a global ratio of $O(d_G)$ on the network latency. Here d_G is the diameter of the network graph and typically smaller than $O(log(M))$, where M is the number of servers in the network.

In the future, we will try this two-stage scheme on different scenarios with different optimization objectives, such as the scenarios that prefer balanced data flow rather than low network latency, to show the expansibility of our two-stage scheme.

References

[1] *Network Functions Virtualization-Introductory White Paper*, ETSI, NFVISG, Sophia Antipolis, France, 2012.

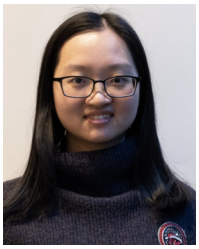
[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[3] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, Aug. 2015.

[4] Z. Guo, S. Liu, and Z.-L. Zhang, "Traffic control for RDMA-enabled data center networks: A survey," *IEEE Syst. J.*, vol. 14, no. 1, pp. 677–688, Mar. 2020.

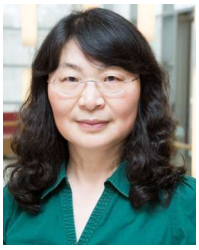
[5] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Nov. 2018.

[6] T. W. Kuo, B. H. Liou, K. C. Lin, and M. J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, Aug. 2018.
[7] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 98–106.
[8] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Service Manag. (CNSM)*, 2015, pp. 50–56.
[9] 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)*, May 2015, pp. 1346–1354.
[10] X. Shang, Z. Liu, and Y. Yang, "Online service function chain placement for cost-effectiveness and network congestion control," *IEEE Trans. Comput.*, vol. 71, no. 1, pp. 27–39, Jan. 2022.
[11] 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 Conf. Comput. Commun.*, Jul. 2020, pp. 267–276.
[12] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.
[13] W. Ma, J. Beltran, D. Pan, and N. Pissinou, "Placing traffic-changing and partially-ordered NFV middleboxes via SDN," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1303–1317, Dec. 2019.
[14] G. Liu, S. Guo, B. Li, and C. Chen, "Joint traffic-aware consolidated middleboxes selection and routing in distributed SDNs," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1415–1429, Jun. 2021.
[15] D. Li, P. Hong, K. Xue, and J. Pei, "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.
[16] J. Liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, "Incremental server deployment for scalable NFV-enabled networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2361–2370.
[17] 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.*, May 2017, pp. 1–9.
[18] 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," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2096–2105.
[19] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 693–701.
[20] 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 Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 1539–1548.
[21] V. Valls, G. Iosifidis, G. D. Mel, and L. Tassiulas, "Online network flow optimization for multi-grade service chains," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1329–1338.
[22] 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 Conf. Comput. Commun.*, May 2022, pp. 170–179.
[23] 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)*, Dec. 2021, pp. 18–25.
[24] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 2071–2084, Oct. 2019.
[25] (2023). *Understand and Use the Enhanced Interior Gateway Routing Protocol (EIGRP)*. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html\#anc17
[26] D. S. Johnson, "*Near-optimal bin packing algorithms*," Dept. Math., Massachusetts Inst. Technol., Cambridge, MA, USA, 1973.
[27] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "Approximate algorithms for the traveling salesperson problem," in *Proc. 15th Annu. Symp. Switching Automata Theory (swat)*, Oct. 1974, pp. 33–42.
[28] M. Kairanbay and H. M. Jani, "A review and evaluations of shortest path algorithms," *Int. J. Sci. Technol. Res.*, vol. 2, no. 6, pp. 99–104, Jun. 2013.
[29] D. Fernholz and V. Ramachandran, "The diameter of sparse random graphs," *Random Struct. Algorithms*, vol. 31, no. 4, pp. 482–516, Dec. 2007.
[30] J. Frey, *Introduction To Stochastic Search and Optimization: Estimation, Simulation, and Control*, vol. 99. Hoboken, NJ, USA: Wiley, 2004, pp. 1204–1205.
[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] M. Daghmehchi Firoozjaei, J. P. Jeong, H. Ko, and H. Kim, "Security challenges with network functions virtualization," *Future Gener. Comput. Syst.*, vol. 67, pp. 315–324, Feb. 2017.



Yingling Mao received the B.S. degree in mathematics and applied mathematics from the Zhiyuan College, Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, software-defined networks, and cloud computing.

Xiaojun Shang received the B.Eng. degree in information science and electronic engineering from Zhejiang University, the M.S. degree in electronic engineering from Columbia University, 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, The University of Texas at Arlington. His research interests include the areas of edge-cloud computing, network virtualization, serverless computing, and machine learning.



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.