

Latency-Sensitive Parallel Multi-Path Service Flow Routing with Segmented VNF Processing in NFV-Enabled Networks

Long Qu, Lingjie Yu, Peng Yu and Maurice Khabbaz

Abstract—In the context of Software Defined Networking (SDN) scenarios, the deployment of multi-path routing has been trending as one of the practical approaches. It serves the two-fold objective of improving the reliability of Service Function Chains (SFCs) and reducing end-to-end delays through parallel processing; this latter being this paper's focal point given it is one of the fundamental objectives of 6G. The literature encloses numerous publications revolving around the exploitation of Virtual Network Function (VNF) duplication and optimal placement to enable parallel processing. However, very little attention has been allocated to segmented VNFs with parallel multi-path data traffic flow routing to catalyze service completion. In reality, the application of segmented task processing is now widely used in our internet life (e.g., real-time video on Youtube). In order to realize the ultra-low end-to-end delay of SFC, we introduce the segmented VNF processing window and implement VNF processing tasks in batches/windows with multi-path routing. Herein, a novel Parallel Multi-Path service flow Routing with processing Windows (PMPRW) scheme is proposed. The PMPRW is formulated as a Mixed Integer Linear Program (MILP), owing to the complexity of which, a Column-Generation (CG) based framework is developed to generate accurate sub-optimal solutions that achieve the same performance as the optimal solution. In order to accelerate the process and enhance the performance, we propose an extended Column Fixing (CF) strategy to help generate new columns in CG. Extensive simulations are conducted to gauge the merit of PMPRW and demonstrate its superiority (as opposed to single-path routing). PMPRW achieves desirable performance by concurrently reducing the overall end-to-end delay (e.g., 22% through parallel dual-path routing).

Index Terms—NFV, SFC scheduling, multi-path routing, processing window, MILP.

I. INTRODUCTION

A. Preliminaries:

Over the past couple of years, a wide variety of newly proliferating verticals and applications (e.g., intelligent and autonomous transportation systems, time-sensitive control of manufacturing processes, ultra-high fidelity media, etc) [1],

L. Qu and L. Yu are with the Faculty of Electrical Engineering and Computer Science of Ningbo University, 315211 Ningbo, China. E-Mail: qulong@nbu.edu.cn and 1332734167@qq.com.

P. Yu is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, 100876 Beijing, China. E-Mail: yupeng@bupt.edu.cn.

M. Khabbaz is with the Computer Science Department of the American University of Beirut, Beirut, Lebanon. E-Mail: maurice.khabbaz@aub.edu.lb.

This work was supported in part by all of: *i*) Ningbo Natural Science Foundation (Grant 2021J070), *ii*) Zhejiang Natural Science Foundation (Grant LY20F010004) and *iii*) National Natural Science Foundation of China (Grant 61801254).

[2] constitute a tangible proof of the continuously evolving challenging requirements (e.g., all-time-anywhere hyper-connectivity, continuous and transparent service availability, larger nodal densities, many-fold the currently achievable data rates, ultra-low latency, etc) [3] that mark the incapability of existing networks to support the adequate functionality and proper operability of emerging network services. For instance, the upsurged need for industrial automation throttling the paradigm shift from Industry 4.0 to Industry X.0 keeps on igniting explosive rises in the number of connected devices that, in turn, propound massive demands for task offloading to edge and/or cloud servers, remote cloud assistance and monitoring, content caching and distribution, security-related services (e.g., FireWall, Intrusion Detection, etc), among others. Concurrently, from an architectural panorama, modern networks are expected to be highly flexible, scalable and capable of dynamically handling service traffic flows with variable transferable data volumes and flow completion deadlines; all these properties having to be realized without compromising the CAPital and OPerational EXpenditures (CAPEX/OPEX) constraints. Facing such stringent challenges, Software-Defined Networking (SDN) and Network Function Virtualization (NFV) present themselves as crucial and prime contributing technologies to unleashing the power and potentials of 5G and Beyond; particularly, 6G, [4]–[6]. First, SDN dissociates the control plane from the data plane. Hence, allowing for network abstraction and, therefore, the incorporation of avant-guard Network Functions (NFs). Then, at this point, NFV comes into play as it disaffiliates specific functions from specialized hardware that, typically, used to be deployed for the purpose of hosting and executing them. Precisely, NFV metamorphoses NFs into plain software known as Virtual Network Functions (VNFs) executable by Virtual Machines (VMs) hosted by one or multiple industry-standard Physical Machines (PMs) (e.g., commodity servers, routers/switches, storage nodes, etc) deployed anywhere within a network, at its edge and/or at end-users' premises, [7]. An adequate VNF placement together with appropriate end-to-end traffic flow routing (e.g., [8]) lead to the arrangement of supported network services into VNF clusters; thus, allowing for: *a*) variegating network development, upgrade and upkeep, *b*) expanding network coverage, *c*) intermixing communication and networking technologies, *d*) reusing, sharing and dynamic fine-grained scaling of network resources to satisfy the capacity requirements of newly incoming service requests.

Aside of the above-elaborated advantages, network soft-

TABLE I
IMPORTANT ACRONYM

Acronym	Meaning
CAPEX	the CAPital EXpenditure
OPEX	the OPerational EXpenditure
SDN	Software-Defined Network
NFV	Network Function Virtualization
NF	Network Function
VNF	Virtual Network Function
VM	Virtual Machine
PM	Physical Machine
MD	Micro Datacenter
SFC	Service Function Chain

varization and function virtualization remain locked under the dome of persisting challenges imposed by the rapid tuck in of next-generation communication networks (*e.g.*, architectural evolution, resource management, on-the-fly VNF instantiation, migration, reliability, performance, among others) coupled with lingering concerns regarding the capability of these aforementioned technologies to realize their reckoned objectives and envisioned results [9], [10]. In this regard, the research community remains continuously active and targeting the reinforcement and solidification of SDN/NFV's utility and indispensable role in catalyzing the deployment of full-fledged next-generation carrier grade networks. The important abbreviations mentioned above are recorded in Table I.

B. Related Work

The work of [11] proposed a scheduling algorithm that only accounted for VNF processing delays. Therein, delays experienced by traffic flows traversing particular routes was completely overlooked. In [12], the authors formulated a complex joint VNF placement and activation scheduling problem for the purpose of optimizing network resources and minimizing delays encountered over virtual routes interconnecting a network service's VNFs. Traffic was scheduled and serially steered from source to destination along single paths traversing VNFs deemed to expose these flows to the minimum feasible processing and transmission delays. A similar problem was also considered by the authors of [13] in the context of mobile core networks. The work of [14] addressed a variant of the VNF placement problem coupled with repetitive migration of VNFs to optimal locations at the edge of a highly dynamic network for the purpose of delay minimization. Nonetheless, therein, it was pretty obvious that the continuous migration of VNFs exerted excessive stress on the network and often incurred significant overheads directly affecting the network's performance, especially under medium-to-high loads. In [15], the authors resolved yet another variant of the VNF placement problem with the objective of only reducing VNF processing delays. Therein, available computing resources pertaining to each node were aggregated in a table and assigned different weights that required periodic updates throughout the network's lifespan. Based on these weights VNFs pertaining to newly admitted network services were assigned to the most

suitable nodes regardless of the bandwidth utilization and therefore delays that may arise from the steering new traffic flows along paths with links close to congestion. This is not to mention the negative impact that this will have on the performance of existing active services hosted by the network. The authors of [12] considered placing VNFs concurrently over the identical VMs in Micro Datacenters (MDs) to avoid inter-MD data traffic exchange. This was especially required since long-distance inter-MD links exhibited considerable propagation delays. Despite their avoidance, these latter delays were counterbalanced by increased per-PM processing delays due splitting that PM's computing power over the hosted VNFs, which, in turn, reduced the per-VNF allocated computing resources; hence, slowing down these VNFs. On a finer granularity, existing VMs with spare resources were targeted to host additional VNFs in [17]. Although this would surely increase the per-VM utilization, it would rather be problematic especially in dynamic environments where the experienced per-VM traffic load is highly variable. In such scenarios, VNFs shall be bound to utilize small amounts of computing power with less flexibility and elasticity to accommodate variable flows. This will inevitably the per-VNF processing delays in such situations. A similar context was considered in [18] while further accounting for VM boot time and VNF installation time with the possibility of deploying multi-threaded VNFs to speed up traffic processing over multi-core VMs. To avoid high operating costs and further minimize delays experienced by network services as their traffic serially flows from one VNF to another along very well defined SFCs, the work of [19] proposes to dynamically instantiate and place additional VNFs or reuse the deployed VNFs to maximize the profits; this, of course, generating new VNFs requires additional expenses such as higher PM utilization, energy consumption and, most importantly, additional resource demands especially in terms of processing power. This is not to mention that, with increased network load and whenever accommodating services with larger VNF chain sizes, delays will surely re-increase; thus, violating E2E deadlines, which, in turn, will cause service embedding failure to re-surface. Reusing existing VNF instances may increase the waiting time of VNFs and increase the overall delay of SFC because multiple VNFs need to share one instance. In addition to the above, numerous other existing work in the literature revolve around similar variants of the above-mentioned VNF placement problems. One common feature among all of these publications is the fact that, to reduce E2E delays, they all focus on reducing processing delays and completely ignore routing delays. This is not to mention that the common trend was to adopt serial data processing, transmission, and routing over single paths traversing well ordered VNFs in a chain-like fashion. Very little work gauged the merit of parallelism when it comes to VNF processing with various requirements such as: *a*) reliability (*e.g.*, [20]), *b*) adaptive resource consumption control and increased coordination to minimize queueing delays (*e.g.*, [21]), *c*) flexible and agile network management (*e.g.*, [22]), *d*) deadline-aware cost-efficient server provisioning (*e.g.*, [23]), among many others. Also, parallel routing has been scarcely investigated such as in [22] where the authors resort to partial

SFC parallelism; that is, only VNFs that serve the purpose of realizing a certain objective (*e.g.*, latency reduction) shall be parallelized if parallelization is feasible. Otherwise, VNFs shall be traversed sequentially in order.

Contemporaneously to parallel processing and routing, SFC and, hence, traffic splitting is a newly emerging paradigm that is recently proving to be quite efficient in improving several critical network QoS metrics. Although, this novel technique is still at its infancy it is receiving significant momentum and is being subject to thorough investigations with the objective of gauging the merit and promoting the utility of softwarization and function/infrastructure virtualization in serving the advent of next-generation networks. The development of technology such as Multi-Path Transmission Control Protocol (MPTCP) [24] also makes traffic splitting possible. In [25], authors successfully implemented QoE-aware video streams by using MPTCP and segment routing in SDN networks. Moreover, the work of [26] targets critical multi-cloud environments and proposed a service deployment design, at the core of which is embedded an SFC splitting mechanism that accounts for the variable individual per-SFC requirements (*e.g.*, bandwidth) and aims at handling latency-critical SFCs much more effectively than existing flow maximization-based approaches. In [27], the authors analyze service reliability of complex SFC configurations (*i.e.*, serial and parallel VNF chaining) and their related backup protection components. Precisely, target SFCs are split into multiple sub-SFCs (an approach that was also followed by [28] where VNFs were replicated to accommodate each subflow) and account for network components' sharing, heterogeneity and component inter-dependency when failures occur. In a similar context, the authors of [29] studied the impact of traffic diversion on the SFC bandwidth utilization.

C. Novel Contributions:

An issue that is commonly identified to the applied SFC-splitting mechanisms in the above-surveyed work is the replication of VNFs (for splitting purposes) over different/newly deployed PMs. Consequently, routing data to these VNFs shall require additional paths with distinct links; thus, incurring an increase in bandwidth utilization. Also, none of these existing publications addresses E2E delay reduction. Both data splitting and traditional VNF scheduling models consider that VNF processing will not start until all data arrives at the node.

Motivated by the existence of the above-identified gaps, this work proposes a novel Multi-Path Routing Mechanism with processing Window (MPRMW) that jointly exploits segmented VNF processing and SFC/traffic splitting with multi-path routing for the purpose of reducing delays. Distinguished from existing work in the literature, herein no additional PMs shall be required for VNF replication and traffic shall traverse multiple paths from one VNF to another in a serial sequence; hence, allowing subsets of VNFs to be hosted by a single PM and, therefore, reduce bandwidth consumption. The processing window allows the data that arrives at the node first in all parallel-enabled paths to be processed directly without waiting for other data. Such data packets are split into multiple batches for processing. Below is an elaboration on the above-mentioned contributions laid out throughout the remaining of

this paper.

This paper's fundamental contributions are as follows:

- 1) Our research proposes a multi-path data transmission technique to realize ultra-low latency HD streaming media. In particular, a segmented VNF task processing method is used to reduce the VNF processing latency. The above scheduling model is formulated and represented by a Mixed Integer Linear Program (MILP) having the objective of minimizing the total E2E delay subject to multiple resource utilization (*e.g.*, bandwidth, node processing capacity, etc).
- 2) Owing to the complexity of the above-formulated MILP (NP-Hard), a Column Generation (CG) based framework with extended Column Fixing (CF) is proposed. Since there is a significant correlation between the columns involving the crowding of PMs when processing VNFs. The existing CF cannot judge the quality of the solution/column. While doing so, it still applies the Greedy algorithm to generate a new solution/column, which cannot ensure that the new solution is feasible. In this paper, we introduce the novel evaluation function related to the objective of guiding column fixing and generating new columns through the extended Pricing Problem (PP) to ensure the high quality of the new columns. In this regard, the column-fixing strategy can significantly reduce the execution time by reducing the solution space size.
- 3) Reported simulation results prove that CG's performance is very close to MILP. It can determine sub-optimal solutions close enough to their optimal counterparts within finite periods. Moreover, our proposed MPRMW outperforms its existing single-path-routing counterparts, notably reducing the experienced E2E delays.

The remaining of this paper is organized as follows. Section II presents the system model and the adopted data center sub-networking scenario. In Section III, the MILP is formulated, and then the CG-based framework is presented in Section IV. Extensive simulations and numerical evaluations are laid out and discussed in Section V. Finally, concluding remarks are presented in Section VI.

II. SYSTEM MODEL AND ILLUSTRATIVE EXAMPLE

A. System Model

The substrate network is represented by a directed graph $G = (N, L)$ where N is the set of PMs and L is the set of directed physical links. Each PM hosts a number of VMs capable of executing VNFs of different types with one VNF being processed at a time. Let I represent the set of network service support requests. Any such request, say $i \in I$ is characterized by a quadruple of parameters $\langle s_i, P_i, n_s(i), n_d(i) \rangle$, where s_i is the required SFC (*i.e.*, the sequence of ordered VNFs needed to support the proper functionality/operability of the target service); that is $s_i = \{f_1^i, f_2^i, f_3^i, \dots, f_j^i\}$ with f_j^i denoting the j^{th} VNF in SFC i . In addition, P_i is the set of packet batches that SFC needs to send. Each SFC contains $|P_i|$ packet batches, where $|P_i|$ represents the length of the set P_i . Each batch of packets consists of multiple packets. The

nodes are divided into source nodes N_s , destination nodes N_d and NFV-enabled nodes N_f (i.e., PM). Therefore, for a request i , its source and destination nodes are $n_s(i), n_d(i)$, respectively. Now, instantiated VNFs hosted by the different PMs consume these PMs' computing resources, which, for simplicity, are quantified in terms of CPU processing capacity C_n (packets/ms) on PM n . Here, the VNF processing will be performed in multiple processing windows containing \tilde{w} packets. The delay for each processing window can be calculated as follows: \tilde{w}/C_n . The VNF processing task of f_j^i is finished once all w_i processing windows have been processed. w_i is obtained from $\lceil |P_i|/\tilde{w} \rceil$, where $\lceil \cdot \rceil$ stands for ceiling function. Besides, when forwarding data from one VNF to another, such data gets transmitted over virtual links mapped into physical paths interconnecting the PMs hosting the interacting VNFs. For instance, suppose data corresponding to network service i is to be transferred from f_j^i hosted by a PM n to VNF f_{j+1}^i , which happens to be hosted by PM k ($n \neq k$). The VNFs are connected by virtual links lv_1 and lv_2 . Also, assume that PMs n and k are connected by a set of directed physical paths $\tilde{P}_k^n = (\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_r)$, where r is the number of paths. Each path consists of multiple physical links. For instance, $\tilde{p}_1 = (n \rightarrow m)$, $\tilde{p}_2 = (n \rightarrow m \rightarrow k)$. Link $(n \rightarrow m)$ has a head node n and a tail node m allowing for the transmission of data from PM n to PM m . As a result, we map virtual links lv_1 and lv_2 into paths \tilde{p}_1 and \tilde{p}_2 . The available bandwidth over link $(n \rightarrow m)$ is B_m^n . At this point, the traffic pertaining to network service i shall be divided into sub-traffic flows and transmitted in parallel over the different available paths between nodes. Therefore, the transmission delay experienced over path \tilde{p}_2 is $|P_2^i| \cdot (B_m^n)^{-1}$, where $|P_2^i|$ is the number of the data packets transmitted over path \tilde{p}_2 . The propagation delay over the link δ_{le} is also considered due to the long distance. Based on this, the total delay experienced as a result of transmitting data over path $\Upsilon_{\tilde{p}_2} = |P_2^i| \cdot (B_m^n)^{-1} + \delta_m^n + \delta_k^m$. Here, note that f_{j+1}^i hosted by node k will only initiate VNF processing once data transmission over one path in \tilde{p}_k^n is successfully completed.

B. Illustrative Example:

In the context of NFV-augmented next-generation software networks required to support massive numbers of Ultra Reliable and Low Latency (URLL) services having exhaustive network resource (e.g., processing/computing power, storage space, link bandwidth, etc) demands, Service Function Chain (SFC) management has been renowned as a significant challenge. This is especially true since all of these network services are expected to be concurrently supported; hence, exerting high network resource utilization to a point where such resources often become scarce and unavailable to accommodate additional incoming services further. Consequently, under such limiting conditions, typical existing joint VNF placement and service scheduling algorithms may fail to allocate adequate resources to host all incoming services in such a way that their pertaining traffic flows meet their end-to-end (E2E) deadlines. Also, under high load, service failures have often been observed due to the inability of existing networks to ensure service sustainability and resource demand

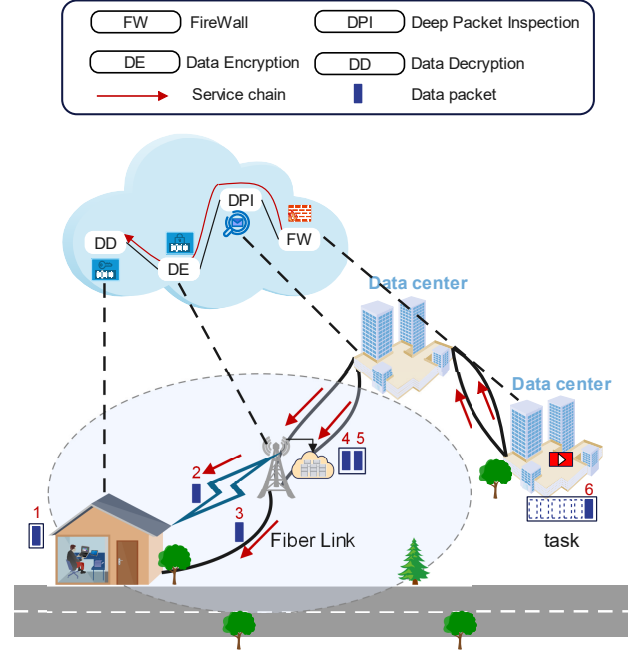
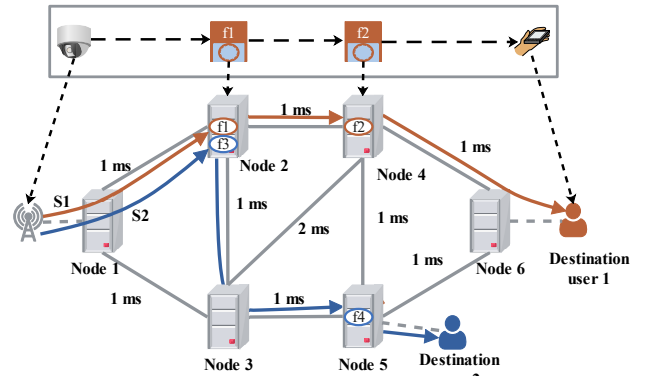
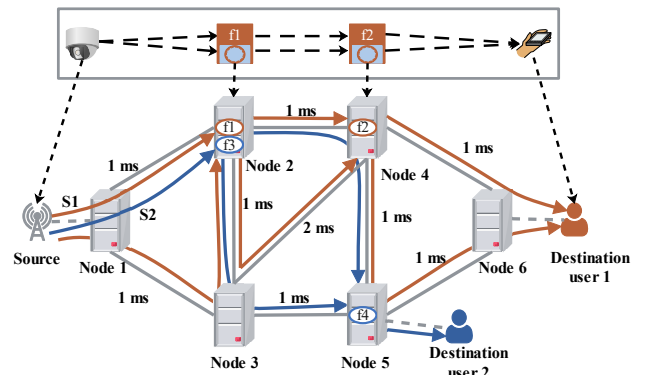


Fig. 1. Illustration of segmented task processing with multi-path routing in a real-time video service



(a) Single-path routing



(b) Multi-path routing

Fig. 2. Illustration of scheduling results of the SFCs

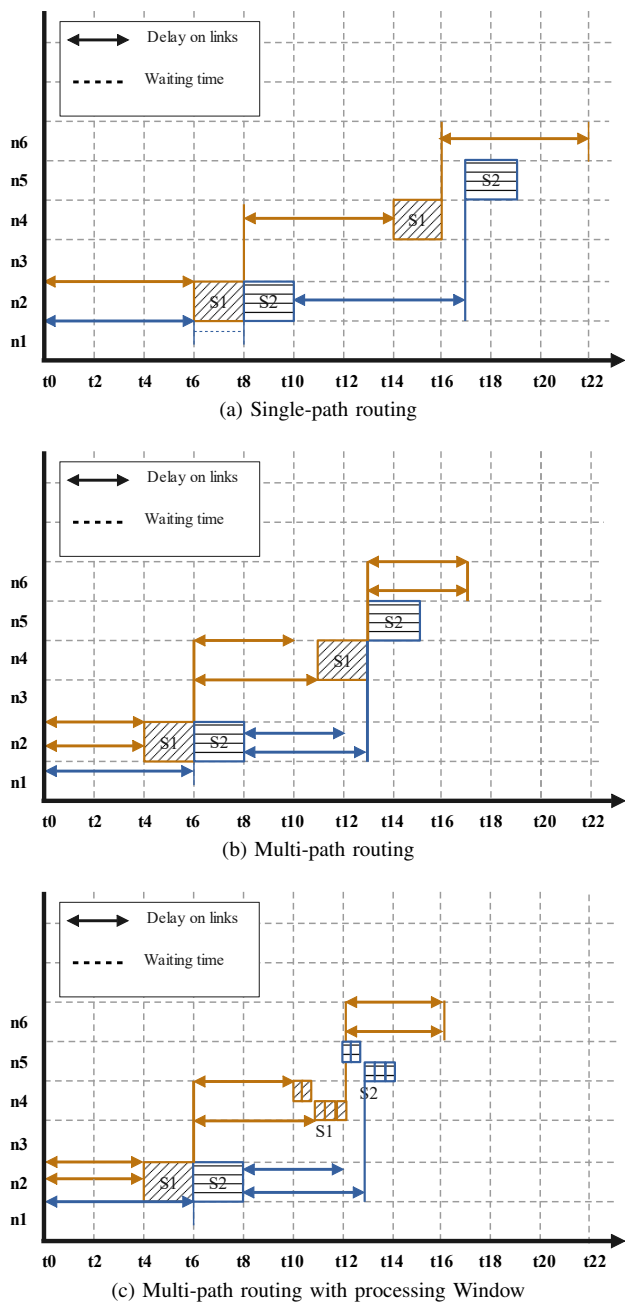


Fig. 3. Illustration of completion time of SFCs

satisfaction; hence, resulting in considerable service drop and blocking probabilities.

This paper proposes our model in an application scenario of the next-generation communication network with HD real-time video streams using segmented task processing. Fig. 1 illustrates a user watching an HD video on YouTube at home. Users only need to receive part of the data (i.e., a batch of data packets) in this video streaming task before they can begin playing. Existing mobile devices often have multiple wireless antennas (e.g., WIFI, 5G), making possible applications such as Multi-Path TCP protocol (MPTCP). Thus, the data of streaming media services can adaptively switch links for transmission depending on the status of the network. Alternately, enabling multiple paths to transmit data in parallel to reduce

end-to-end delay. We provide this service in the NFV-enabled network. The video streaming task contains a batch of data packages (p_1, p_2, \dots, p_6) . Multiple parallel paths are activated when transferring data between VNFs. In the NFV-enabled network, providing a service requires multiple VNFs. First, the service must pass through the FireWall (FW) to ensure that the request complies with the security standards. Subsequently, the traffic is imported into the Deep Packet Inspection (DPI) function to identify whether the service is legal and then managed through bandwidth restriction, priority adjustment, etc. When the service satisfies the functions above, the video stream will be encoded (decoded) through data encryption (decryption) and played on the user's device. End-to-end delay is crucial for HD streaming media. Although MPTCP may help reduce service latency in the future. However, when the VMs process VNFs, it still needs to ensure that it receives a complete batch of data, which may lead to a negative impact of VNF processing tasks on the advantages of MPTCP. Our work focuses on service latency by using segmented VNF task processing methods to reduce the delay of VNF processing. As shown in Fig. 1, the data center transmits data packets to users. Packets (p_1, \dots, p_5) are transferred on the network. There are multiple optical fibers between data centers. So we transmit packets parallelly on them. The edge cloud center hosts VNF DE. After receiving packets (p_4, p_5) , we start VNF processing tasks in the edge cloud center. Data packets (p_2, p_3) have completed DE and are being sent to the user through 5G and optical fiber. At this time, the user receives packet p_1 and starts to process the DD task without waiting for the complete arrival of this batch of packets. The VNF processing tasks are segmented on each hosted center. As a result, we realize the ultra-low latency HD streaming media.

This section is dedicated to providing an illustrative example to highlight and clarify the effectiveness of SFC splitting and parallel routing with processing windows in reducing the overall per-SFC experienced E2E delay. Consider the application scenario where a mobile user (e.g., smartphone or tablet) connects to a home smart appliance (e.g. a surveillance camera) for real-time monitoring and activity control. Consequently, data shall be exchanged between these two ends constituting the source and destination nodes communicating remotely. This service is supported through a Data Center Network (DCN). In this context, data traffic flowing back and forth between these two nodes is required to traverse an ordered sequence of VNFs making up the service's SFC. Each of that service's VNFs is mapped to/hosted by a PM within the DCN substrate, as shown in Fig. 2. There are two SFCs in the network, $s_1 = (f_1, f_2)$ and $s_2 = (f_3, f_4)$. The SFC mapping process can be summarized as follows: placing VNFs on PMs that can support them and mapping virtual links between VNFs to physical paths in the network. There is only one virtual link between VNFs in Fig. 2a. For Fig. 2b, the number of virtual links between VNFs increases to 2, which enables virtual links to be mapped to various physical paths. However, this does not mean that the two virtual paths must be mapped to two different physical paths. Since the SFC embedding method for SFCs is similar, here we only show the mapping process for s_1 . The propagation

delay on each link is indicated in Fig. 2. Additionally, we only consider the latency of data transmission in the DCN, and the latency of the upload (download) access network $source \rightarrow n_1$ ($n_6 \rightarrow user1, n_5 \rightarrow user2$) is ignored. It is assumed that the transferred packets of SFC s_1 are $P_1 = (p_1^1, p_2^1, \dots, p_5^1)$ and s_2 's transferred packets are $P_2 = (p_1^2, p_2^2, \dots, p_5^2)$. The per-packet VNF processing time and transmission delay is 0.4 ms and 1 ms, respectively. For a physical path composed of multiple links such as ($n_2 \rightarrow n_3 \rightarrow n_5$), n_3 is the relay node along path, which forwards data from n_2 to n_5 . Forwarding data is assumed not to consume any nodal resources and instantaneously executed (*i.e.*, with zero forwarding time). In Fig. 3a, we introduce the traditional SFC scheduling model with single-path routing. The delay of packets on path $\tilde{p}_1 = n_1 \rightarrow n_2$ is 5×1 ms (transmission delay) + 1 ms (propagation delay). Therefore, we can easily calculate the completion time of s_1 ($22 = 6 \times 3 + 0.4 \times 5 \times 2$) and s_2 ($17 = 6 + 7 + 0.4 \times 5 \times 2$). However, the data from s_1 and s_2 arrive at n_2 at 6 ms. Since the node can only process one VNF at a time, f_3 must wait for 2 ms until f_1 is finished. As a result, s_2 's completion time is delayed by 2 ms to 19 ms. In Fig. 3b, We employ multi-path routing to reduce the delay of data on the link. PM must still receive all data packets before processing VNFs as in Fig. 3a. Since s_1 enables the multi-path routing method to speed up the data transfer process, the packets are split into two parts when transferred from n_1 to f_1 . For data transmission from n_1 to n_2 , paths $\tilde{p}_1 = n_1 \rightarrow n_2$ and $\tilde{p}_2 = n_1 \rightarrow n_3 \rightarrow n_2$ are concurrently enabled. We use path \tilde{p}_1 (1 hop) to transfer packets (p_1^1, p_2^1, p_3^1) and path \tilde{p}_2 (2 hops) to transfer (p_4^1, p_5^1) . Thus, the delay on paths can be calculated as $\Upsilon_{\tilde{p}_1} = 1 \times 3 + 1 = 4$ ms, $\Upsilon_{\tilde{p}_2} = 1 \times 2 + 1 + 1 = 4$ ms. As a result, f_1 's processing begins 2ms earlier and concludes at 6 ms. Then n_2 returns to the idle state, and all data packets of s_2 reach node n_2 , so it can start VNF processing for s_2 immediately. After processing f_1 , s_1 is transmitted from f_1 to f_2 . The data is divided into two parts: packets (p_1^1, p_2^1, p_3^1) on $\tilde{p}_3 = (n_2 \rightarrow n_4)$ and packets (p_4^1, p_5^1) on $\tilde{p}_4 = (n_2 \rightarrow n_3 \rightarrow n_4)$. Although it takes only 4 ms for the data on \tilde{p}_3 to reach node n_4 , the node will still hold off on processing until the data packets on \tilde{p}_4 arrive 1 ms later. Similarly, paths $n_4 \rightarrow n_6$ (p_1^1, p_2^1, p_3^1) and $n_4 \rightarrow n_5 \rightarrow n_6$ (p_4^1, p_5^1) are also enabled for transferring data from n_4 to n_6 leading to a delay reduction by 2 ms as well. So the completion time of s_1 is 17 ($4 + 2 + 5 + 2 + 4$) ms and s_2 15 ($6 + 2 + 5 + 2$) ms. In order to utilize the delay gap between multiple paths to process some packets in advance to reduce the VNF processing delay, Fig. 3c illustrates how we segment VNF processing tasks using processing windows based on the multi-path routing in Fig. 3b. The length of windows is 1 packet. As a result, VNF processing tasks (f_2, f_4) are segmented. The processing window can first process packets (p_1^1, p_2^1) when the data on \tilde{p}_1 reaches (10 ms). At the 10.8 ($10 + 0.4 \times 2$) ms, packets (p_1^1, p_2^1) are completed. So we can begin processing as soon as data packets (p_3^1, p_4^1, p_5^1) arrives at the node at 11 ms. Finally, f_2 completes the processing at 12.2 ($11 + 0.4 \times 3$) ms (*i.e.*, a reduction of 0.8 ms compared to the scenario in Fig. 3b). In order to maintain data integrity, the processing windows, in this case, still needs to process VNFs

TABLE II
KEY VARIABLES AND THEIR SIGNIFICANCE.

Variable	Description
$x_n^{i,j} = \{0; 1\}$	Indicates whether VNF f_j^i is hosted by PM n
$y_{j,le}^{i,lv} = \{0; 1\}$	Indicates if virtual link lv is mapped to physical link le as f_j^i 's descending data pathway
$\varphi_{j,le}^{i,lv} = \{0; 1\}$	Indicates whether data on virtual link lv is uploaded to the network by physical link le , where the head node of le is the node hosting VNF f_j^i
$\gamma_{j,lv}^{i,p} = \{0; 1\}$	Indicates whether the p-th packet of service s_i is allocated on virtual link lv after completing the processing of VNF f_j^i
$\alpha_{j,le}^{i,lv}$	The integer variable represents the size of data allocated on virtual link lv as the downlink of VNF f_j^i , which is mapped to the physical link le
$v_{j,le}^i$	The continuous variable represents the bandwidth allocated by SFC s_i on physical link le
$\beta_{lv}^{i,j} = \{0; 1\}$	Indicates whether data is assigned to virtual link lv as a downlink to VNF f_j^i
$\tau_{j,lv}^i$	The continuous variable represents the delay on virtual link lv
$t_j^{i,p}$	The continuous variable represents the p-th packet arriving at the node where VNF f_j^i is placed
te_j^i	The continuous variable represents the completion time of VNF f_j^i
$ts_{j,w}^i$	The time when the w-th window of VNF f_j^i starts processing
$\tilde{x}_{u,v}^{i,j} = \{0; 1\}$	Indicates VNFs f_j^i and f_v^u from different services placed on the same PM, with f_j^i starting to process data before f_v^u .
$\tilde{x}_{i,j}^{u,v} = \{0; 1\}$	Indicates VNFs f_j^i and f_v^u from different services placed on the same PM, with f_v^u starting to process data before f_j^i .
\tilde{te}_i	The continuous variable represents the completion time of SFC s_i
Parameter	Description
$\tilde{s}_n^i, (\tilde{d}_n^i) = \{0; 1\}$	The binary parameter indicates whether PM n is the source (destination) node of service s_i
δ_{le}, B_{le}	The bandwidth capacity (propagation delay) of physical link le
C_n	The CPU processing capacity of PM n
\tilde{w}	The length of processing windows

in the order of data packets. Thus, the total experienced delay for s_1 is reduced from 22 ms (single-path routing) to 17 ms (Fig. 3b) and 16.2 ms ($4 + 2 + 4 + 1 + 1.2 + 4$ in Fig. 3c).

III. PROBLEM FORMULATION

The next section is dedicated for formulating our proposed problem. The relevant variables used hereafter and their significance is listed in Table II.

Eq. (1) ensures that a VNF can be only hosted on one PM. Moreover, VNFs can only be installed on nodes that can support them in Eq. (2).

$$\sum_{n \in N} x_n^{i,j} = 1 \quad (1)$$

$$x_n^{i,j} \leq 0, \forall n \in N/N_f \quad (2)$$

After VNF placement, we introduce the flow balance constraint in this paper. To ensure that if VNF f_j^i is hosted on node n , service will pass through that node, Eq. (3) must be satisfied. Data on each physical link $le = (n, n')$ flows from the head node n to the tail node n' . In particular, we define the physical link whose head (tail) node is n as the downlink (uplink) of n .

$$\sum_{le.head=n} y_{j,le}^{i,lv} \geq x_n^{i,j}, \forall i \in I, f_j^i \in s_i, le \in L, lv \quad (3)$$

Eq. (4) ensures that service can be routed from the source to the destination through a series of nodes where VNFs are placed. If f_j^i and f_{j+1}^i are hosted on PM n and n' , the virtual link lv between them should be mapped to a series of continuous physical links from n to n' . When $j = 0$, lv transmits data from the source node to node n where the first VNF is placed.

$$\begin{aligned} & \sum_{le.head=n} y_{j,le}^{i,lv} - \sum_{le.tail=n} y_{j,le}^{i,lv} \\ &= \begin{cases} x_n^{i,j} - x_n^{i,j+1} & \text{otherwise, } j \in (1, \dots, |s_i| - 1) \\ \tilde{s}_n^i - x_n^{i,j+1} & \text{if } n = n_s(i), j = 0 \\ x_n^{i,j} - \tilde{d}_n^i & \text{if } n = n_d(i), j = |s_i| \end{cases} \quad (4) \end{aligned}$$

In Eq. (5), we introduce $\varphi_{j,le}^{i,lv}$ to represent the outgoing ports of NFV-enabled node n . The downlink of node n is used to symbolize these ports. L_j^i is the set of downlink virtual links of VNF f_j^i .

$$\varphi_{j,le}^{i,lv} = \begin{cases} y_{j,le}^{i,lv} \times x_n^{i,j} & \text{otherwise, } \forall f_j^i \in s_i, n \in N_f \\ y_{j,le}^{i,lv} \times \tilde{s}_n^i & \text{if } j = 0, n = n_s(i) \end{cases} \quad (5)$$

Because Eq. (5) is nonlinear, we use the following methods to linearize it. We only show the linearization of the first equation. The linearization of the other equations can be done similarly.

$$\begin{aligned} & \forall i \in I, lv \in L_j^i, f_j^i \in s_i, le.head = n \\ & \varphi_{j,le}^{i,lv} \leq y_{j,le}^{i,lv} \quad (6) \end{aligned}$$

$$\varphi_{j,le}^{i,lv} \leq x_n^{i,j} \quad (7)$$

$$\varphi_{j,le}^{i,lv} \geq x_n^{i,j} + y_{j,le}^{i,lv} - 1 \quad (8)$$

Eq. (9) ensures that VNF f_{j+1}^i or $n_d(i)$ receives s_i 's complete data, which have been split and transmitted in parallel over different paths. In Eq. (10), at least one packet is transmitted through each virtual link. The minimum data fraction is 1 packet.

$$\sum_{lv \in L_j^i} \gamma_{j,lv}^{i,p} = 1, \forall i \in I, j \in (0, \dots, |s_i|), p \in P_i \quad (9)$$

$$\sum_{p \in P_i} \gamma_{j,lv}^{i,p} \geq 1, \forall i \in I, j \in (0, \dots, |s_i|), lv \in L_j^i \quad (10)$$

Eq. (11) calculates the size of data sent over the mapped physical path by the virtual link. It can be linearized in a

similar way as Eq. (5). As data is distributed for simultaneous transmission across parallel virtual links, the required bandwidth for service s_i is correspondingly distributed in Eq. (12). Finally, all bandwidth allocated on a link needs to be less than the bandwidth capacity of that link in Eq. (13).

$$\begin{aligned} & \forall i \in I, lv \in L_j^i, j \in (0, \dots, |s_i|), le.head = n \\ & \alpha_{j,le}^{i,lv} = \sum_{p \in P_i} \gamma_{j,lv}^{i,p} \times y_{j,le}^{i,lv} \quad (11) \end{aligned}$$

$$v_{j,le}^i = \frac{\sum_{lv \in L_j^i} \alpha_{j,le}^{i,lv}}{|P_i|} \times v_i \quad (12)$$

$$\sum_{i \in I} \sum_{j \in (0, \dots, |s_i|)} v_{j,le}^i \leq B_{le} \quad (13)$$

Here we consider that the delay of data transmission over the link $\tau_{j,lv}^i$ consists of two components, transmission delay, and propagation delay. The first term on the right side of Eq. (14) represents the transmission delay on lv . It is calculated by dividing the total size of data sent out from a given outgoing port by the bandwidth of the downlink to that port. δ_{le} is a fixed constant representing the propagation delay on the link.

$$\begin{aligned} \tau_{j,lv}^i & \geq \frac{\sum_{lv \in L_j^i} \alpha_{j,le}^{i,lv} \times \varphi_{j,le}^{i,lv}}{B_{le}} + \sum_{le \in L} y_{j,le}^{i,lv} \times \delta_{le} \\ & \forall i \in I, j \in (0, \dots, |s_i|) \quad (14) \end{aligned}$$

For each data packet, $t_j^{i,p}$ ($t_{|P_i|+1}^{i,p}$) represents its arrival time on the PM hosting f_j^i (destination). Eq. (15) ensures the orderly processing of data in SFC. When the data packet has completed processing the previous VNF, it can be transmitted to the node hosting the following VNF through the physical path. Particularly, te_0^i represents the time when SFC s_i starts transmission in the network.

$$\begin{aligned} & \forall i \in I, j \in (1, \dots, |s_i| - 1), lv \in L_j^i, p \in P_i \\ & t_{j+1}^{i,p} \geq te_j^i + \tau_{j,lv}^i \times y_{j,lv}^{i,p} \quad (15) \end{aligned}$$

To ensure data integrity, for the current VNF processing window, VNF processing can only start when both the packets to be processed in the current window and the previous packets have arrived at the node in Eq. (16). Eq. (17) indicates the order of processing between windows. According to $|P_i| - (w_i - 1) \times \tilde{w}$, how many packets need to be processed in the last processing window can be calculated. Finally, we calculate the completion time of VNF te_j^i in (18).

$$ts_{j,w}^i \geq t_j^{i,p}, \forall p \in [1, \dots, \min(\tilde{w} \times w, |P_i|)] \quad (16)$$

$$ts_{j,w+1}^i \geq ts_{j,w}^i + \sum_{n \in N} x_n^{i,j} \times \frac{\tilde{w}}{C_n}, \forall w \in [1, \dots, w_i - 1] \quad (17)$$

$$te_j^i \geq t_{j,w_i}^i + \sum_{n \in N} \frac{|P_i| - (w_i - 1) \times \tilde{w}}{C_n} \times x_n^{i,j} \quad (18)$$

For VNFs hosted on the same PM, the PM cannot process these VNFs simultaneously. Therefore, we determine the processing order of these VNFs by Eqs (19)-(20).

$$\forall f_j^i, f_v^u, f_j^i \neq f_v^u, n \in N_f$$

$$x_n^{i,j} + x_{v,n}^u - 1 \leq \tilde{x}_{u,v}^{i,j} + \tilde{x}_{i,j}^{u,v} \leq 1 \quad (19)$$

$$te_j^i \leq ts_{v,1}^u + (1 - \tilde{x}_{u,v}^{i,j}) \times M \quad (20)$$

The main objective of this present optimization problem is to minimize the total completion times of all SFCs, which is defined as:

$$\tilde{te}_i \geq t_{|s_i|+1}^{i,p} \quad (21)$$

$$K = \sum_{i \in I} \tilde{te}_i \quad (22)$$

SFC finishes once all of its data packets reach the destination node.

Therefore, this problem's MILP formulation boils down to:

$$\min K \text{ s.t. (1) - (22)} \quad (23)$$

IV. COLUMN GENERATION ALGORITHM

It is very well known (refer to [30]) that the joint VNF placement and scheduling problem is already NP-Hard. In this work, the complexity is even further increased when jointly considering data splitting and parallel routing with processing windows. In this sense, the above-formulated problem is also NP-hard as the time complexity for resolving this problem, indeed, grows exponentially as a function of the number of service requests. Although the problem's resolution may be feasible within a finite (though large) time interval for small-scale networks, it becomes almost non-feasible for medium-to-large-scale scenarios. In large-scale networks, conventional heuristic algorithms are prone to getting stuck in local optima [19]. Meanwhile, reinforcement learning may fail to converge when facing such complex problems [31]. Under such circumstances, CG has proven to be highly efficient in generating sub-optimal solutions that are very close to their optimal counterparts.

Compared with MILP, CG can significantly reduce execution time. We can regard the solution obtained by CG as a lower bound of the MILP optimal solution [32]. The CG divides the problem into two parts, the restricted main problem (RMP) and the pricing problem (PP). The optimal solution to the dual problem of MILP can be obtained by solving the RMP. In this paper, we create a PP for each SFC to generate new columns. Then the dual variables of the relevant constraints are passed to the PP. The objective of PP is to minimize the Reduced Cost (RC). If $RC < 0$, we generate new columns to optimize the main problem further. We introduce an extended CF and modify the PP to reduce the number of iterations considering the relationship between columns. The newly generated columns are returned to RMP. The two problems constantly exchange information and find the optimal solution after multiple iterations.

A. Restricted Main Problem

1) Parameters:

- \tilde{te}_c^i : represents the completion time corresponding to the choice of column c for SFC s_i .
- $o_{c,n}^{i,t} = (0; 1)$: indicates that PM n is processing the VNFs of column c for service s_i at time slot t .
- $\tilde{v}_{c,le}^i$: represents the bandwidth allocated on link le in column c of service s_i .

2) Variables:

Let $\rho_c^i = (0; 1) := 1$ if for service s_i , column c is selected. As such, RMP is formulated as:

$$\min \sum_{i \in I} \sum_{c \in C} \tilde{te}_c^i \times \rho_c^i \quad (24)$$

subject to:

$$\sum_{c \in C} \rho_c^i = 1, \forall i \in I \quad (25)$$

$$\sum_{i \in I} \sum_{c \in C} o_{c,n}^{i,t} \times \rho_c^i \leq 1, \forall t \in T, n \in N \quad (26)$$

$$\sum_{i \in I} \sum_{c \in C} \tilde{v}_{c,le}^i \times \rho_c^i \leq B_{le}, \forall le \in L \quad (27)$$

To obtain the dual variables of the constraints associated with the RMP, ρ_c^i is relaxed to a continuous variable with the range of [0-1]. Eq. (25) ensures that for each SFC s_i , at least one column will be selected. For PM n , multiple VNFs cannot be processed at the same time in (26). Eq. (27) indicates that the total bandwidth allocated on each physical link cannot exceed the bandwidth capacity of that link. Thereby, the RMP is modeled as an LP problem that minimizes the sum of the completion times of all SFCs and satisfies constraints (25-27). $\theta, \bar{\theta}_i, \hat{\theta}_t^n, \check{\theta}_{le}$ are dual variables associated with Eqs. (24-27) respectively.

B. Pricing Problem

For each SFC, we generate a sub pricing problem. The constraints in each pricing problem are similar to the original problem in Section III. For a particular PP of SFC s_i , the relevant constraints can be formulated as follows:

$$\sum_{j \in F_i} \sum_{n \in N} x_n^j = 1 \quad (28)$$

$$x_n^j \leq 0, \forall j \in F_i, n \in N/N_f \quad (29)$$

$$\sum_{le.head=n} y_{lv,le}^j \geq x_n^j, \forall j \in F_i, le, lv \quad (30)$$

$$\sum_{le.head=n} y_{lv,le}^j - \sum_{le.tail=n} y_{lv,le}^j = \begin{cases} x_n^j - x_n^{j+1} & \text{otherwise, } j \in (1, \dots, |F_i| - 1) \\ \tilde{s}_n^i - x_n^{j+1} & \text{if } n = n_s(i), j = 0 \\ x_n^j - \tilde{d}_n^i & \text{if } n = n_d(i), j = |F_i| \end{cases} \quad (31)$$

$$\varphi_{lv,le}^j = \begin{cases} y_{lv,le}^j \times x_n^j & \text{otherwise, } \forall j \in F_i, n \in N_f \\ y_{lv,le}^j \times \tilde{s}_n^i & \text{if } j = 0, n = n_s(i) \end{cases} \quad (32)$$

$\forall lv \in L_j^i, le.head = n$

$$\sum_{lv \in L_j^i} \gamma_{j,lv}^p = 1, \forall j \in F_i, p \in P_i \quad (33)$$

$$\alpha_{lv,le}^j = \sum_{p \in P_i} \gamma_{j,lv}^p \times y_{lv,le}^j \quad (34)$$

$$v_{le}^j = \frac{\sum_{lv \in L_j^i} \alpha_{lv,le}^j}{|P_i|} \times v_i \quad (35)$$

$$\sum_{j \in F_i} v_{le}^j \leq B_{le} \quad (36)$$

$$\tau_{lv}^j \geq \frac{\sum_{lv \in L_j^i} \alpha_{lv,le}^j \times \varphi_{lv,le}^j}{B_{le}} + \sum_{le \in L} y_{lv,le}^j \times \delta_{le} \quad (37)$$

$$t_{j+1}^p \geq t_{ej} + \tau_{lv}^j \times y_{lv,le}^j, \forall j, lv, p \quad (38)$$

$$ts_w^j \geq t_j^p, \forall p \in [1, \dots, \min(\tilde{w} \times w, |P_i|)] \quad (39)$$

$$ts_{w+1}^j \geq ts_w^j + \sum_n x_n^j \times \frac{\tilde{w}}{C_n}, \forall w \in [1, \dots, w_i - 1] \quad (40)$$

$$te_j \geq t_{w_i}^j + \sum_{n \in N} \frac{|P_i| - (w_i - 1) \times \tilde{w}}{C_n} \times x_n^j \quad (41)$$

$$\tilde{te} \geq te_{|F_i|+1}^p \quad (42)$$

$$\tilde{K} = \theta \times \tilde{te} - \tilde{\theta}_{le} \times \sum_{j \in F_i} v_{le}^j - \bar{\theta}_i \quad (43)$$

Eqs. (28-29) are VNF placement constraints. Eqs. (30-31) represent the flow constraints. Eqs. (32-36) define the size of data and bandwidth allocated on the physical paths. Eq. (37) calculates the delay of data on the links and is deployed in Eq. (38) to ensure that VNFs are processed sequentially. In Eqs. (39-42), we list the constraints related to VNF processing. As demonstrated in Eq. (43), the objective of PP is to minimize the reduced cost. When \tilde{K} is negative, PP will be instructed to generate new columns for each SFC and return the relevant parameters to RMP. RMP only needs to select the optimal combination of columns, thus significantly reducing the execution time of the problem. Here, each PP is a continuous time model, and to satisfy (26) in RMP, we can calculate the value of $\rho_{c,n}^{i,t}$ by t_j^p, te_j , thus dividing the time into T time slots in RMP. Similarly, $\tilde{v}_{c,le}^i$ (\tilde{te}_c^i) is also obtained from v_{le}^j (\tilde{te}).

The dual variables $\hat{\theta}_i^n$ are not returned to the PP problem. However, the corresponding constraint (26) is essential in RMP and will directly affect the combination of columns. In [33], the authors proposed a column fixing strategy with backtracking for fair working time in airline crew scheduling Problem. For the first time, we apply this strategy to the SFC scheduling issue. The existing column-fixing policy still generates new columns using the Greedy algorithm. At the same time, there needs to be an effective evaluation method for the fixed columns selected in the RMP. So we extend the column-fixing policy in this paper. The parameters of fixed columns are passed to PP to ensure the feasibility of new columns. Therefore, we add new constraints in PP to meet the column-fixing policy. Secondly, we propose an evaluation

function to fix columns. The benefits of the extended column-fixing strategy are: 1) Increasing the number of columns in the CG algorithm to improve performance. 2) Improving the quality of newly generated columns and reducing the algorithm's execution time. 3) Taking into account the relationship between the columns of different SFCs. We will next describe in detail the column-fixing strategy used in this paper.

C. Column-fixing Strategy

Suppose the PP problem's objective is converted to the completion time. So it is clear that such a scheduling strategy is optimal for a single SFC. We define the set of columns constituted by this optimal scheduling policy as $C^* = (c_1^*, \dots, c_i^*)$. The set of corresponding optimal completion times is $Te^* = (te_1^*, \dots, te_i^*)$. After RMP is solved, we can obtain the value of ρ_c^i for the current iteration round. We use $\rho_{c,k}^i$ to represent the selected column c for SFC s_i in the k -th iteration. Since $\rho_{c,k}^i$ is relaxed into a continuous variable, for each SFC, we will select the column with the biggest value of $\rho_{c,k}^i$ (i.e., $\rho_{c,k}^i = \max(\rho_{c,k}^i, \forall c \in C)$). In each iteration, we fix some columns from all selected columns by selected criteria, where the set consisting of fixed columns is denoted by C_k . The set of SFCs corresponding to C_k is denoted by S_k . If a column is fixed, then we will not add new columns for its corresponding SFC. We provide two criteria to select fixed columns as follows: 1) The column is selected as a fixed column if satisfying $\tilde{te}_c^i - te_i^* < \pi$, where \tilde{te}_c^i is the completion time of column c of SFC s_i . 2) Fixing partial columns according to the evaluation function (44). Θ_i^k of the k -th iteration is updated using the value of the previous iteration and \tilde{te}_c^i associated with the selected column in RMP. λ is the weight factor. In (45), we take the completion time corresponding to the optimal column as the initial value of Θ_i^k . Eventually, if $|\tilde{te}_c^i \times \rho_{c,k}^i - \Theta_i^k| < \pi$, the column is fixed.

$$\Theta_i^k = \Theta_i^{k-1} + \lambda \times (\tilde{te}_c^i - \Theta_i^{k-1}) \quad (44)$$

$$\Theta_i^1 = te_i^* \quad (45)$$

When solving PPs for other SFCs of columns to be updated, the parameters of the fixed columns will be sent to these PPs. As a result, we modify the initial PPs by adding the following constraints.

1) Parameters:

$z_{v,n}^u = (0; 1), u \neq i$: The binary parameter indicates that the VNF f_v^u of SFC s_u is hosted on node n , if the column corresponding to this SFC is in the fixed column set.

$ts_v^u (te_v^u), u \neq i$: The continuous parameter represents the completion (start) time of VNF f_v^u in the fixed column

2) Variables:

$z_{u,v}^j = (0; 1), u \neq i$: $z_{u,v}^j$ represents the VNF of the current SFC and other SFCs if placed on the same node, the VNF of the current SFC is processed first.

$z_j^{u,v} = (0; 1), u \neq i$: $z_j^{u,v}$ For VNFs placed on the same node, the VNF f_v^u in other SFCs starts processing before the current SFC.

$$z_n^j + z_{v,n}^u - 1 \leq z_{u,v}^j + z_j^{u,v} \leq 1 \quad (46)$$

$$te_j \leq ts_v^u + z_j^{u,v} \times M, \forall j, n, u \in S_k, v \in F_u \quad (47)$$

$$te_v^u \leq ts_1^j + z_{u,v}^j \times M, \forall j, n, u \in S_k, v \in F_u \quad (48)$$

Constraints (46-48) ensure that nodes cannot handle multiple VNFs at the same time. We next describe the proposed CG based framework in Algorithm 1.

Algorithm 1 CG based framework

Input: G, I

Output: ρ_c^i, te_c^i

```

1: Generate initial set of columns  $C$  using Greedy algorithm
2: while  $k < k_{max}$  do
3:   Solve RMP
4:   if  $k == 1$  then
5:     Solve PP for each SFC and add new column in  $C$ 
6:     Initialize  $C^*, Te^*$ 
7:   else
8:     (update  $\Theta_i^k$  for selected criterion 2)
9:     Fix part columns according to selected criterion
10:    if  $S_k == I$  then
11:      break
12:    end if
13:    for  $s_i \in I/S_k$  do
14:      Solve PP after adding new constraints (46-48)
15:      If the objective is negative, add new column to  $C$ 
16:      and calculate the value of  $o_{c,n}^{i,t}, \tilde{v}_{c,le}^i$ .
17:    end for
18:    if all SFCs in  $I/S_k$  return positive objective values
19:    then
20:      break
21:    end if
22:  end while
23: return  $\rho_c^i, te_c^i$  and calculate  $\tilde{K}$ 

```

First, the initial C is generated by using Greedy algorithm. We first obtain the shortest path from the source to the destination node. Only 1 virtual link is enabled between VNFs. VNFs are placed along the shortest path, and all virtual links are mapped to the physical links along the shortest path. The length of the processing window satisfies $\tilde{w} > \max(|P_i|, \forall i \in I)$. Based on the mapping results, the SFC scheduling is finished in the real time slots. Subsequently, we solve RMP and obtain the dual values of the associated constraints (line 3). We solve the original PP in the first iteration to obtain the optimal scheduling results for each SFC without considering the interactions between SFCs (lines 5-6). We then proceed to the next iteration, and in all subsequent iterations, we fix the partial columns according to the selected guideline. For criterion 2, we need to update its evaluation function in each iteration (line 8). From line 9, we get the fixed columns C_k and the associated parameters: $S_k, z_{j,n}^i, ts_j^i, te_j^i$. Constraints (46-48) and the parameters of fixed columns will be added to the PP of the remaining SFCs. New columns are generated for these SFCs by solving PPs. The new column will be added to C only if the subjective of PP is negative (lines 13-16). The algorithm stops when all remaining SFCs return positive

values (lines 17-19), or all columns of SFCs are fixed (lines 10-12).

D. Algorithm Complexity Analysis:

The MILP is converted to RMP and PP in CG-based framework. Since CG is an iterative algorithm, we cannot determine the number of iterations. So we only calculate the complexity of CG in each iteration. The RMP is an ILP problem with few variables, so the execution time of CG mainly depends on the complexity of the PPs. In the first iteration round, we run the PP without enabling the column fixing strategy. The dominant variables are $\varphi_{lv,le}^j, y_{lv,le}^j, \gamma_{j,lv}^p, \alpha_{lv,le}^j$. The corresponding main constraints are (31), (34) with complexity of $O(|P| \times |J| \times |L_v| \times |L|)$ and $O(|J| \times |L_v| \times |L| \times |N|)$. Here $|L_v|$ represents the maximum number of virtual paths allowed to be enabled between VNFs. $|P|$ is the number of data packets and $|J|$ is the number of VNFs. Then we enable CF and add new constraints to the PP with the same complexity. Therefore, the complexity of each iteration is $O(|J| \times |L_v| \times |L| \times (|P| + |N|))$.

V. PERFORMANCE EVALUATION

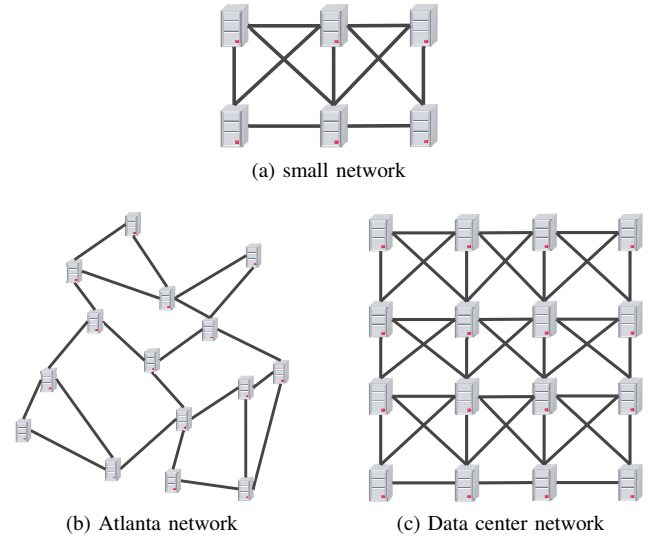


Fig. 4. Networks in Evaluation

In this section, the performance of the proposed algorithm is evaluated. The algorithm is executed on a server with Intel Xeon Gold 5220R @2.20GHz, 256GB RAM, Windows 10. The optimal solution is generated by Gurobi 9.5 in Python 3.8. The networks used are shown in Fig. 4. Each link contains two directional links with opposite directions. We first test the proposed model in a small network (6 nodes, 11 links). Then the proposed model is extended to the Atlanta network with 15 nodes and 22 links [34] and a data center network with 16 nodes and 42 links. Table III lists the value of relevant parameters in detail.

A. Comparison of MILP and CG

We first compare the performance of MILP and CG based frameworks in a small network, which is indicated in Fig. 4a. For all purposes of simplicity, the symbol $|L_v|$ shall be used in

TABLE III
RELEVANT PARAMETERS

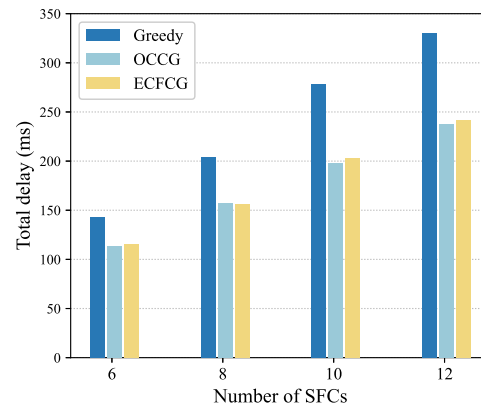
Definition	Parameters	Value
Bandwidth of physical links	B_{le}	10 Mbps
CPU processing capacity	C_n	[0.5-2] packets/ms
Length of processing window	\tilde{w}	2, [1-5] (packets)
Propagation delay	δ_{le}	1 ms
Transmission delay per packet		1 ms
Set of packets per SFC	P_i	$ P_i =3-6$
Set of VNFs per SFC	F_i	$ F_i =1-2$

TABLE IV
SIMULATION RESULT

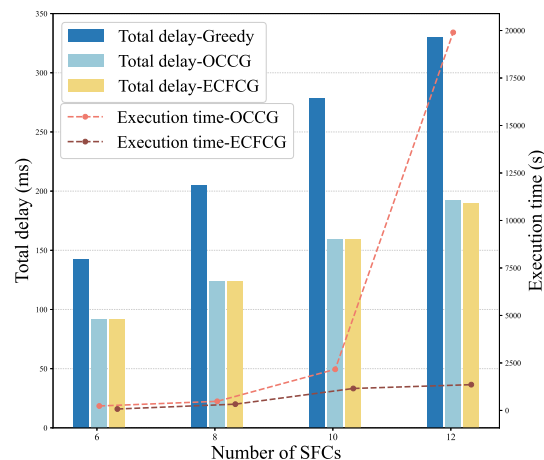
Algorithm	$ L_v $	Number of SFCs	Total delay (ms)	Execution time (s)
MILP	1	[1, 2, 3]	[12, 27.5, 48.5]	[0.05, 0.32, 3]
	2	[1, 2, 3]	[9, 22, -]	[0.3, 1272, ∞]
	3	[1, 2, 3]	[8, 20, 35]	[0.43, 19, 2852]
OCCG	1	[1, 2, 3]	[12, 27.5, 48.5]	[0.1, 0.84, 1]
	2	[1, 2, 3]	[9, 22, 38]	[0.57, 3, 7]
	3	[1, 2, 3]	[8, 20, 35]	[1, 15.8, 20]
ECFCG	1	[1, 2, 3]	[12, 27.5, 48.5]	[0.1, 0.47, 0.55]
	2	[1, 2, 3]	[9, 22, 38]	[0.34, 2, 3]
	3	[1, 2, 3]	[8, 20, 35]	[1, 10, 10.2]

the sequel to represent the maximum number of parallel virtual links allowed for data transmission between VNFs. We call the CG-based framework that use different criteria to fix columns: 1) Optimality-fixed Column (OCCG) algorithm 2) Evaluation Function-based Column Fixing (ECFCG) algorithm. The values of λ and π are 0.05 and 1. As shown in Table IV, both CG-based frameworks have the same performance as MILP. It is evident that as $|I|$ increases to 3, the MILP's execution time increases exponentially when multi-path parallel routing is enabled. For single-path routing, MILP only takes 0.43 s, while the execution time increases rapidly to 2852 s for three-path routing. At this time, CG can achieve the same performance in only a dozen seconds. In most cases, ECFCG can obtain solutions faster compared with OCCG. We can also find that the total latency is decreased by about 22% and 30%, respectively, when $|L_v| = 2$ and $|L_v| = 3$, compared with $|L_v| = 1$. This result demonstrates the proposed model's superiority to the traditional SFC scheduling model.

Next, the performance of the proposed algorithms is compared to that of a Greedy algorithm in Fig. 5. The Greedy algorithm generates the initial solution for CG in Section IV. We cannot solve MILP to obtain the optimal solution in a feasible time. Therefore, only the results of CG-based frameworks are shown in Fig. 5. The performance of CG frameworks is far better than that of Greedy method. This demonstrates that CG efficiently optimize the total delay of SFCs while adding additional columns during iterations. Compared with the Greedy algorithm, when only single path routing is enabled, the two algorithms reduce the total delay of 25% (OCCG) and 24% (ECFCG), respectively. In 5b, both algorithms reduce the total delay by at least 35%. The reduction rises to 42% as the number of SFCs grows. Although



(a) The total delay of SFCs with single-path routing



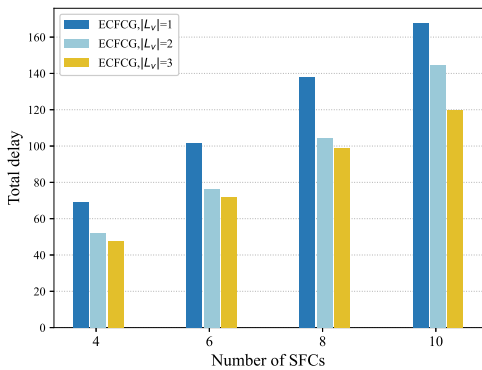
(b) The comparison of performance and execution time of different algorithms when $|L_v| = 2$

Fig. 5. The performance of algorithms in the Atlanta network

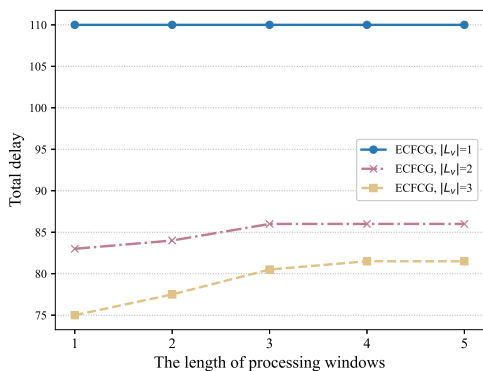
the performance of OCCG and ECFCG is almost the same, ECFCG's execution time is less than OCCG's. In particular, when there are 12 SFCs, ECFCG solves it in about 23 minutes, while OCCG takes 5.5 hours. This is because ECFCG not only uses the optimal scheduling column for a single SFC but also uses the parameters of the column selected in the previous iteration to update the evaluation function, which can help the CG algorithm optimize toward the objective. The OCCG only uses the historical optimal scheduling column of each SFC. Since ECFCG can obtain the almost same performance as OCCG in less time, we will run only ECFCG in the subsequent evaluations.

B. Performance evaluation of ECFCG

In this section, we compare the performance performance of ECFCG in different $|L_v|$. We test the scenario when $|L_v| = 3$ in a data center network we created because there are not enough links in the Atlanta network. When $|L_v| = 3$, its latency is significantly reduced compared to $|L_v| = 2$ in Fig. 6a. We use $|L_v| = 1$ as a benchmark and calculate the average latency reduction by 22% and 29% when $|L_v| = 2$ and $|L_v| = 3$, respectively. In this paper, we propose a parallel



(a) The total delay of SFCs



(b) Effect of processing window length on performance

Fig. 6. (a) The performance of ECFCG in data center network. (b) The performance of ECFCG with different length of processing windows.

multi-path routing strategy and use processing windows for segmented VNF processing tasks. In Fig. 6b, we change the number of packets processed by the processing windows simultaneously (i.e., the window length \tilde{w}). When $|L_v| = 1$, changing the value of \tilde{w} has no impact on the performance. However, when $|L_v| \geq 2$, the total delay increases with increasing \tilde{w} and eventually remains stable. As discussed in Section II, VNF processing tasks can be segmented using processing windows in combination with multi-path routing strategies to reduce latency. However, the performance is equal to conventional VNF processing when the processing window is long enough. It is not difficult to discover that the first arriving packets must wait for more packets to meet the window length. The node cannot begin processing VNFs until it receives packets that fit the window length. Therefore, the segmented VNF processing tasks cannot utilize the delay difference across parallel paths to reduce the VNF processing delay with a long window length.

As illustrated in Fig. 7, with the increase in the time required to process per packet, the gap between multi-path and single-path routing becomes smaller. Compared to single path routing, the reduction of delay achieved by ECFCG- $|L_v|=2$ decreases from 25% to 0.03%. The performance gap between $|L_v| = 2$ and $|L_v| = 3$ also becomes smaller. This demonstrates how challenging it is to enable more parallel virtual links to counteract the adverse effects of increasing VNF processing time on overall latency. It shows that the

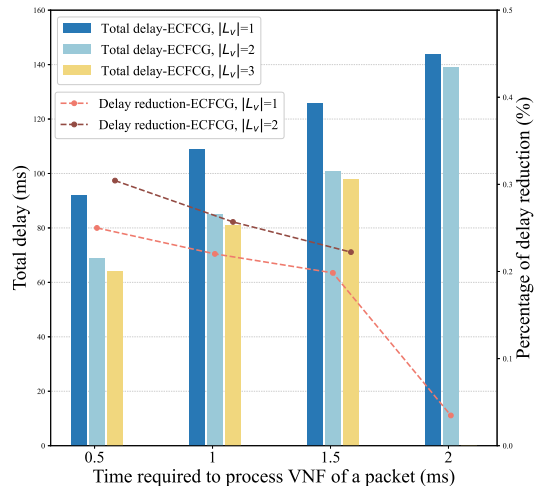


Fig. 7. The performance of ECFCG in data center network

MPRMW can reduce the delay more effectively when the main factor of SFC delay is data transmission latency on the links.

VI. CONCLUSION

This paper proposed a novel Multi-path Parallel routing Mechanism with processing Window (MPRMW) for NFV-enabled SDNs. The joint batch VNF placement and parallel SFC routing problem was formulated as an MILP, owing to the complexity of which a CG-based framework with CF was proposed to generate highly accurate sub-optimal solutions within minimal time intervals. In comparison with a Greedy algorithm and MILP in the context of various scenarios, ECFCG's superiority was proven in terms of various critical performance metrics. In particular it is observed that the number of parallel paths directly affect the SFC latency. Compared to existing single-path transmission methods, enabling two paths can reduce end-to-end latency by 22%, and further, enabling three paths can reduce end-to-end latency by 30%. Similarly, if the processing window length exceeds a certain value, the processing delay of VNFs cannot be reduced.

Motivated by such a finding, the effect of the number of parallel paths and the length of processing windows on the achieved ECFCG performance was investigated in the context of different scenarios. It is not difficult to find that MPRMW is applicable to those service chains where the transmission delay accounts for a high proportion of the overall service delay. On the basis of the multi-path routing, selecting a smaller processing window can further reduce the VNF processing delay.

REFERENCES

- [1] N. Maleki, A. Musaddiq and D. Toll, "DynaSens: Dynamic Scheduling for IoT Devices Sustainability," *CoBCoM*, 2022, pp. 1-7.
- [2] A. Elewah, W. M. Ibrahim and A. Rafikl, "ThingsDriver: A Unified Interoperable Driver for IoT Nodes," *IWCMC*, 2022, pp. 877-882.
- [3] B. Karg and S. Lucia, "Towards low-energy, low-cost and high-performance IoT-based operation of interconnected systems," *Proc. IEEE WF-IoT*, 2018, pp. 706-711.

- [4] N. M. Nasir, S. Hassan and K. M. Zaini, "Evolution Towards 6G Intelligent Wireless Networks: The Motivations and Challenges on the Enabling Technologies," *Proc. IEEE SCORED*, 2021, pp. 305-310.
- [5] M. S. Akbar, (et. al.), "6G Survey On Challenges, Requirements, Applications, Key Enabling Technologies, Use Cases, AI Integration Issues and Security Aspects," *arXiv*, 2022, ONLINE, URL: <https://arxiv.org/abs/2206.00868>
- [6] A. Kak, "Towards 6G Through SDN and NFV-Based Solutions For Terrestrial And Non-Terrestrial Networks," *Ph.D. Thesis, Georgia Institute of Technology*, 2021.
- [7] R. A. Eichelberger, T. Ferreto, S. Tandel and P. A. P. R. Duarte, "SFC Path Tracer: A Troubleshooting Tool For Service Function Chaining," *Proc. IFIP/IEEE IM*, 2017, pp. 568-571.
- [8] L. Qu, M. Khabbaz and C. Assi, "Reliability-Aware Service Chaining In Carrier-Grade Softwarized Networks," *IEEE JSAC*, vol. 36, no. 3, pp. 558-573, March 2018.
- [9] Partha Pratim Ray and Neeraj Kumar, "SDN/NFV Architectures For Edge-Cloud Oriented IoT: A Systematic Review," *ELSEVIER, Comp. Comm.*, Vol. 169, pp. 558-573, 2021.
- [10] P. V. Mekikis, K. Ramantas and A. Antonopoulos, "NFV-Enabled Experimental Platform for 5G Tactile Internet Support in Industrial Environments," *Proc. IEEE Trans Industr Inform*, 2020, pp. 1895-1903.
- [11] J. Li, W. Shi, N. Zhang and X. S. Shen, "Reinforcement Learning Based VNF Scheduling with End-to-End Delay Guarantee," *ICCC*, 2019, pp. 572-577.
- [12] L. Qu, C. Assi and K. Shaban, "Delay-Aware Scheduling and Resource Optimization With Network Function Virtualization," *IEEE Trans. Commun*, vol. 64, no. 9, pp. 3746-3758, Sept. 2016.
- [13] B. Ren, S. Gu, D. Guo, G. Tang and X. Lin, "Joint Optimization of VNF Placement and Flow Scheduling in Mobile Core Network," *IEEE TCC*, Early Access, vol. 10, no. 3, pp. 1900-1912, 1 July-Sept. 2022.
- [14] R. Cziva, C. Anagnostopoulos and D. P. Pezaros, "Dynamic, Latency-Optimal VNF Placement At The Network Edge," *Proc. IEEE INFOCOM*, 2018, pp. 693-701.
- [15] S. I. Kim and H. S. Kim, "A VNF Placement Method Based on VNF Characteristics," *Proc. IEEE ICOIN*, 2021, pp. 864-869.
- [16] T. Wang, J. Zu, G. Hu and D. Peng, "Adaptive Service Function Chain Scheduling In Mobile Edge Computing Via Deep Reinforcement Learning," *IEEE Access*, Vol. 8, pp. 164922-164935, 2020.
- [17] Z. Wang, J. Zhang, T. Huang and Y. Liu, "Service Function Chain Composition, Placement, and Assignment in Data Centers," *IEEE TNSM*, vol. 16, no. 4, pp. 1638-1650, Dec. 2019.
- [18] T. Gao (et. al.), "Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks," *IEEE TCOM*, vol. 68, no. 8, pp. 4946-4959, Aug. 2020.
- [19] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho and N. Crespi, "Joint VNF Placement and Scheduling for Latency-sensitive Services," *IEEE T-NSE*, vol. 9, no. 4, pp. 2432-2449, 1 July-Aug. 2022.
- [20] M. Huang, W. Liang, X. Shen, Y. Ma and H. Kan, "Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing," *IEEE TMC*, vol. 19, no. 11, pp. 2699-2713, 1 Nov. 2020.
- [21] J. Cai, Z. Huang, L. Liao, J. Luo and W.-X. Liu, "APPM: Adaptive Parallel Processing Mechanism For Service Function Chains," *IEEE TNSM*, vol. 18, no. 2, pp. 1540-1555, June 2021.
- [22] I. -C. Lin, Y. -H. Yeh and K. C. -J. Lin, "Toward Optimal Partial Parallelization for Service Function Chaining," *IEEE/ACM T-NET*, vol. 29, no. 5, pp. 2033-2044, Oct. 2021.
- [23] S. Kianpisheh and R. H. Glitho, "Cost-Efficient Server Provisioning for Deadline-Constrained VNFs Chains: A Parallel VNF Processing Approach," *Proc. IEEE CCNC*, 2019, pp. 1-6.
- [24] A. A. Barakabitze, L. Sun, I. -H. Mkwawa and E. Ifeachor, "A Novel QoE-Centric SDN-Based Multipath Routing Approach for Multimedia Services over 5G Networks," *ICC*, 2018, pp. 1-7.
- [25] A. A. Barakabitze, I. -H. Mkwawa, L. Sun and E. Ifeachor, "QualitySDN: Improving Video Quality using MPTCP and Segment Routing in SDN/NFV," *NetSoft*, 2018, pp. 182-186.
- [26] P. Hegyi, "Service Deployment Design in Latency-Critical Multi-Cloud Environment," *ELSEVIER Comp. Net.*, 2022, In Press, ONLINE: <https://www.sciencedirect.com/science/article/pii/S1389128622001487>
- [27] Engelmann, Anna, and Admela Jukan, "A Simple Reliability Analysis of Complex Service Function Chains (SFCs)," *arXiv*, 2020, ONLINE: <https://arxiv.org/pdf/2005.02495.pdf>.
- [28] N. Promwongsa (et. al.), "Ensuring Reliability and Low Cost When Using a Parallel VNF Processing Approach to Embed Delay-Constrained Slices," *IEEE TNSM*, vol. 17, no. 4, pp. 2226-2241, Dec. 2020.
- [29] Beck, Michael Till and Botero, Juan Felipe, "Coordinated Allocation of Service Function Chains," *Proc. IEEE GLOBECOM*, 2015, pp. 1-6.
- [30] L. Liu, S. Guo, G. Liu and Y. Yang, "Joint Dynamical VNF Placement and SFC Routing in NFV-Enabled SDNs," *IEEE TNSM*, vol. 18, no. 4, pp. 4263-4276, Dec. 2021.
- [31] Y. -H. Hsu, T. -R. Tsai, T. -C. Yeh and Y. -L. Wang, "Deep Reinforcement Learning based Mobility-aware SFC Embedding for MEC in 5G and Beyond," *IEEE WCNC*, 2023, pp. 1-6.
- [32] H. A. Alameddine, S. Sebbah and C. Assi, "On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach," *IEEE TNSM*, vol. 14, no. 4, pp. 860-874, Dec. 2017.
- [33] Y. Iijima (et. al.), "Column generation heuristics to airline crew scheduling problem for fair working time," *Proc. IEEE SMC*, 2016, pp. 003217-003222.
- [34] T. Sato, A. Kikuchi, R. Shinkuma and E. Oki, "Column Generation Based Algorithm for Service Chaining Relaxing Visit Order and Routing Constraints," *GLOBECOM*, 2020, pp. 1-6.