On Efficient VNF-FG Design in IoT Networks

Cheng Ren, Jiangping Zhang, Yu Wang, and Yaxin Li

***Abstract*—In recent times, it has been witnessed that an increasing number of Internet connected devices impose an huge challenge on Internet of Thing (IoT) networks, which provides diverse and complex network services through edge computing empowered IoT terminals. A network service can be formally represented by Virtualized Network Function Forwarding Graph (VNF-FG) with the advent of Network Function Virtualization (NFV) technology. Previous researches mainly focus on VNF-FG embedding (VNF-FGE) and take VNF-FG as the input. In this paper, we investigate the design of VNF-FG, which is required to be a Directed Acyclic Graph (DAG) and achieved by the IoT terminal, in two scenarios. In static scenario, for a set of traffic flows arriving at the IoT terminal and requesting different network services, an ILP model P_s including loop prevention constraints is well formulated. An approximation algorithm AFGC with competitive ratio $O(1+(|R|-1)\sigma), \sigma \in (0,1)$ is then designed, which thoroughly search all key instances to run comprehensive loop break. In dynamic scenario for a flow request on the fly reaching an IoT terminal, an ILP model P_d and another approximation algorithm DFGU with a competitive ratio $O(1+\frac{|SC_r|}{|F_r|})$ are developed, giving priority to reuse of existing topology to generate an augmented VNF-FG of minimum size. Simulation results indicate AFGC and DFGU outperform state-of-art work, and the gap between each of the two algorithms and their respective ILP models is marginal.**

***Index Terms*—Internet of Things, directed acyclic graph, virtualized network function forwarding graph design.**

I. Introduction

With the explosive growth of Internet connected devices and widespread popularity of B5G/6G technique, IoT networks, that create connectivity of anything from anywhere, face huge challenges in terms of impressive flow scales and diverse and complex network service requests [1], [2]. With the support of edge computing (EC), the IoT terminal serves as edge computing gateway (the two terms are used indiscriminately in this paper) in the EC-IoT framework shown in Fig. 1, which is capable of processing local data on the connected devices [3]. The data flows received by the IoT terminal via rich IoT interfaces, are originally generated by connected devices that are associated with certain applications installed on the IoT terminal, and further sent to cloud IoT platform through Internet to obtain specific network services. The cloud IoT platform refers to the cloud service IoT platform provided by service provider, which takes VNF-FG as input to implement service provisioning.

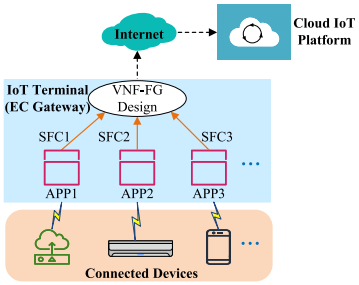


Fig. 1. EC-IoT framework.

Service requirement of the data flow can be represented by an ordered sequence of VNFs called a service function chain (SFC) [4], [5], which is served by service provider through edge computing gateway. However, with the increase of number of devices connected to IoT terminals, providing services at the granularity of SFC to individual flows becomes less efficient for IoT network and may lead to user experience deterioration. The IoT terminal is envisioned to collect and combine the SFCs required by different flows to generate a VNF-FG. Compared with SFC which is a linear chain of VNFs, VNF-FG comes with graph structure characteristic and is required to be a DAG [6], [7]. ETSI advocates the use of VNF-FG to abstract complex network services, which further draws attention to VNF-FG design. However, most existing researches focus on VNF-FGE which directly takes VNF-FG as the input, rather than fundamental design of VNF-FG [8], [9].

In this paper, we explore problem of VNF-FG design, which acts as an ancestor step and of great importance to resource allocation [10]. The essence of VNF-FG design is to generate a directed acyclic virtual graph of minimum size, which requires minimum number of virtual nodes (or instances, the two terms are used indiscriminately) for each VNF type. The indepth study on VNF-FG design raises three fundamental challenges. First, given a set of SFCs, initializing a single instance for each VNF type is generally not enough as loops may appear. Then, it is crucial to identify key instances in the loop. Key instance that results in the loop is typically shared by multiple flows and needs to be replicated to break the loop. Note that, not all instances shared by multiple flows are key instances. Second, when multiple loops coexist, how to determine the order to perform loop break is of vital importance, since several loops may nested and different order may greatly affect

Manuscript received 30 January 2024; accepted 1 April 2024. Date of publication 24 April 2024; date of current version 21 August 2024. The associate editor coordinating the review of this article and approving it for publication was J.-F. Botero. *(Corresponding author: Cheng Ren.)*

Cheng Ren is with the School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China, and also with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: rencheng@swpu.edu.cn).

Jiangping Zhang, Yu Wang, and Yaxin Li are with the School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China.

Digital Object Identifier 10.1109/TNSM.2024.3392976

1932-4537 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

the efficiency. Third, when key instances are confirmed, how to determine the number of copies of key instances necessary to form an acyclic graph is critical, as minimum number of instances is preferred.

In particular, we study VNF-FG design under static and dynamic scenarios, considering a set of SFCs of different flow requests reaching an IoT terminal and a single SFC on the fly respectively. The main contributions are summarized as follows:

- We prove the NP-hardness of VNF-FG design, and formulate it as an ILP model $\boldsymbol{P}_s$ in static scenario to obtain optimal solutions. Especially, constraints to guarantee loop free of the resulting VNF-FG are literally constructed for the first time.
- A standard loop elimination procedure with three steps are proposed, based on which, an efficient acyclic forwarding graph construction (AFGC) approximation algorithm with proved competitive ratio is developed.
- In dynamic scenario, ILP model $\boldsymbol{P}_d$ is formulated, which particularly concerns the overlap between the new incoming service chain and existing VNF-FG. Then, a dynamic forwarding graph updating (DFGU) approximation algorithm with proved competitive ratio is designed, whereby new virtual links are categorized to perform different combination operation sequentially.
- Extensive simulation experiments have been carried out. The results show that the performance of AFGC and DFGU is better than state-of-art work. The gap between each of the two proposed algorithms and the corresponding ILP model is marginal.

The rest of the paper is organized as follows: Section II discusses related work. Section III gives problem definition. Section IV presents $\boldsymbol{P}_s$ and AFGC algorithm for static scenario. Section V details $\boldsymbol{P}_d$ and DFGU algorithm in dynamic scenario. Section VI analysis simulation performance. We draw conclusions in Section VII.

II. Related Work

With ever-increasing data traffic generated in IoT networks, issues related to NFV, coupled with SDN, have attracted widespread interest [11], [12]. Previous work mainly focuses on VNF placement in physical network, where network service is simply represented as an SFC [13], [14]. Ye et al. in [15] first propose concept of VNF combination, and deal with joint topology design and mapping of SFCs. However, SFC topology design refers to combination of different VNFs in the same SFC, which can be seen as a part of VNF placement and thus is essentially different from VNF-FG design. Authors in [16] propose a heuristic to design chaining of SFC considering ratios of outgoing to ingoing flows. However, it is not suitable to design VNF-FG which has much complex structure.

Recent studies consider a more complex service abstraction model, VNF-FG, and mainly embed VNF-FG into substrate network known as VNF-FGE [17], rather than elaborately investigating how to obtain the graph structure of VNF-FG, namely the VNF-FG design [18]. Authors in [19] propose an algorithm based on Genetic Algorithm to deploy traffic control network functions to promote end-to-end IoT traffic control. Reference [20] investigates VNF-FGE in dynamic IoT networks by combining Graph Neural Network (GNN) with Deep Reinforcement Learning (DRL). Reference [21] also aims at the problem of VNF-FGE by proposing algorithms based on GCN and DRL. In [22], VNF chaining requests are represented as connected graphs, but the details of VNF forwarding graph construction is neglected. In [23], a novel eigendecomposition based approach is proposed, and two use cases are discussed referring to VNF-FGE rather than VNF-FG design.

The closest work to ours is [24] and [25]. In [24], Cao et al. propose a two-step method composed of flow designing and flow combining to generate a VNF-FG. The technique for flow designing is similar to that of [16]. For flow combining, an instance is simply determined as a key instance if it is traversed by multiple flows. And the required number of replicas equals to the number of flows passing through this instance. This may result in redundant replicas in the VNF-FG. Furthermore, when it comes to complex situations, such as the occurrence of more loops, more flows passing through the loops, and more key instances that may coexist in the loops, this is still an open research area that needs to be thoroughly investigated. And this is exactly what we plan to do in this paper. Reference [25] proposes VNF-FG extension algorithms considering new service chains and VNF types. However, it directly performs VNF-FG scaling without optimization and focuses on resource allocation essentially. In summary, very few work has been done on how to achieve optimal VNG-FG, which is a challenging problem and, as a previous step, very important for VNG-FGE.

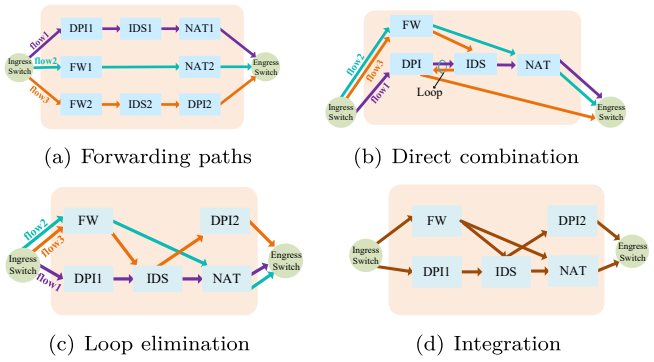
III. Problem Definition

Given a set of SFCs collected by an IoT terminal, we assume a dummy ingress and egress switch to facilitate VNF-FG design. Each traffic flow of an application needs to traverse a specific ordered sequence of VNFs between two dummy switches, to form a virtual network forwarding path (NFP). Let $NFP = \{NFP_1, \ldots, NFP_k\}$, in which each element NFP_i corresponds to $flow_i$. k indexes the total number of flows reaching the IoT terminal. The VNF-FG is constructed by IoT terminal through appropriate combination of all the forwarding paths. In particular, we divide the process of VNF-FG design into three phases and use a simple example to illustrate it.

(i) *Direct consolidation*. Merge virtual nodes of identical VNF type to obtain a direct consolidation graph. For example, direct consolidation is performed on three flows in Fig. 2(a), to obtain Fig. 2(b). A loop appears between *DPI* and *IDS*, which reveals the fact that direct consolidation is not enough.

(ii) *Loop elimination*. In each loop, identify key instances and duplicate them. In this example, apparently, either of *DPI* and *IDS* in the loop shared by two flows can be key instance. We can duplicate either of them into 2 copies to break the loop. In Fig. 2(c), *DPI* is duplicated into DPI_1 and DPI_2, which are for $flow_1$ and $flow_3$ respectively, to achieve loop break.

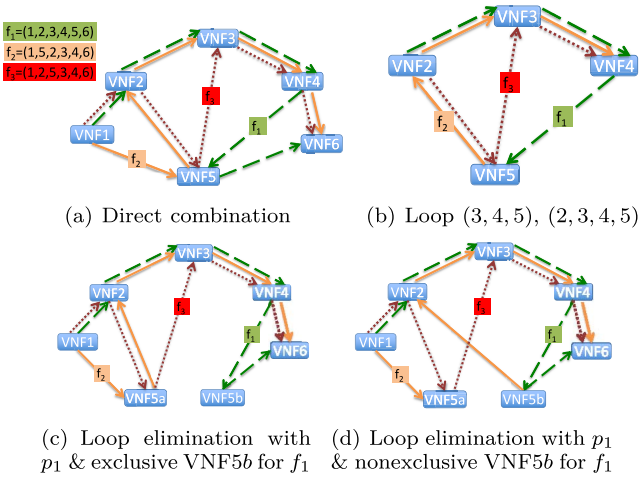
Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.



(a) Forwarding paths (b) Direct combination

(c) Loop elimination (d) Integration

Fig. 2. Simple example of VNF-FG design.



(a) Direct combination (b) Loop $(3, 4, 5)$, $(2, 3, 4, 5)$

(c) Loop elimination with p_1 & exclusive VNF5b for f_1 (d) Loop elimination with p_1 & nonexclusive VNF5b for f_1

Fig. 3. Example of loop elimination procedure.

(iii) *Integration*. Merge links connecting same nodes to regulate virtual topology shown in Fig. 2(d).

Summarily, when SFCs of different flows have identical VNFs, they share the same VNF instances in a directed virtual graph, loops may appear. To eliminate loops in a directed graph, let some VNF instances in the loop shared by multiple flows be no longer shared. The VNF instances that could not be shared are so-called key instances, which need to be replicated into several copies and monopolized by one or more flows on the loop. Furthermore, in complicate situations with multiple loops and multiple nodes in the loop, to determine fewest key instances and fewest replicas is of vital importance to *loop elimination*. Therefore, a general procedure for *loop elimination* is drawn out with an example for illustration right after.

Standard ***loop elimination procedure*** regarding loop check and break includes 3 steps.

• *Step 1*. Find all loops in direct consolidation graph with depth first search (DFS) algorithm. For overlapped loops, start with the bigger one.

• *Step 2*. In a target loop, search all *loop paths* and find the paths with maximum hops. *loop path* is defined as a path of a flow in the loop. For the longest *loop paths*, let the one carrying minimum number of flows on its last hop be the target *loop path*. Otherwise, pick one randomly.

• *Step 3*. On a target *loop path*, explore backward starting from its last VNF (e.g., VNF_l). Calculate the number of flows denoted by n that go through the last hop $\text{VNF}_l \rightarrow \text{VNF}_{l-1}$, make each of these flows monopolize one instance of VNF_l exclusively and all other flows share another instance. If $n > 1$, investigate VNF_{l-1} similarly. Otherwise, output current topology.

Example: In Fig. 3(a), flow f_1, f_2 and f_3 arrive with predefined SFCs $(e.g., f_1 = (1, 2, 3, 4, 5, 6))$. Three virtual paths denoted by different lines are directly combined, which results in loop (3, 4, 5) and (2, 3, 4, 5). *Step 1*: start with the bigger one, loop (2, 3, 4, 5), as it is more likely to break the two in one shot. *Step 2*: identify *loop paths* $p_1 : 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$, $p_2 : 5 \rightarrow 2 \rightarrow 3 \rightarrow 4$, $p_3 : 3 \rightarrow 4$ and pick the longest *loop paths*. For p_1 and p_2 with identical length, p_1 has a single flow f_1 on its last hop $4 \rightarrow 5$. p_2 has three flows on its last hop $3 \rightarrow 4$, which requires 3 instances of VNF4 and further exploration for VNF3. Obviously, *loop elimination* based on p_1 is better. *Step 3*: On p_1, the last node VNF5 is traversed only by f_1, which means 2 instances VNF5a and VNF5b instead of VNF5 are necessary to break the loop. Specifically, VNF5b is monopolized by f_1, and the other flows are designated to VNF5a as shown in Fig. 3(c). Otherwise, loop (2, 3, 4, 5b) still exists as shown in Fig. 3(d). As only f_1 passes though the last hop, no further investigation is required on preceding nodes of p_1.

Given an instance of the VNF orchestration problem (VNF-OP) which is proved to be NP-hard in [26], we can transform it to an instance of VNF-FG with following two additional requirements. (i) VNFs of different types could only be map to different nodes. (ii) After aggregating the directed routing paths of all SFCs by merging links between identical nodes, no loops should occur. If we relax the two constraints, then VNF-FG can be reduced to VNF-OP. Hence, VNF-FG is NP-hard as well.

IV. VNF-FG Design in Static Scenario

In static scenario, a group of flows with a given set of SFCs arrive at an IoT terminal. A validation method for directed acyclic graph is proposed to help formulate an ILP model to achieve optimal VNF-FG. Then, as VNF-FG design is proved to be NP-hard, an approximation algorithm is further developed placing emphasis on ***loop elimination procedure*** proposed in Section III.

Lemma 1: For a VNF-FG $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$, we repeatedly prune all the terminal nodes and the links connected to them. After $|\boldsymbol{V}|$ iterations, if no nodes or links are left, $\boldsymbol{G}$ is acyclic, otherwise, $\boldsymbol{G}$ is cyclic.

Proof: When there is a loop, nodes on the loop must have both incoming and out-going links. Nodes without incoming or out-going links, namely terminal nodes, do not form loops. In this regard, based on $\boldsymbol{G}$, repeatedly delete all terminal nodes and the links connected to them. Each time all terminal nodes and affiliated links in current graph are pruned, new terminal nodes may appear, which makes the maximum iterations be $|\boldsymbol{V}|$. If $\boldsymbol{G}$ is acyclic, it will be completely prune after $|\boldsymbol{V}|$

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

TABLE I
NOTATION FOR PROBLEM $\boldsymbol{P}_s$

Parameter	Description
$\boldsymbol{V}$	The set of candidate virtual nodes, $\boldsymbol{V}=\{u\}$.
k_u^j	A binary parameter indicating whether virtual node u represents VNF j.
$\boldsymbol{R}$	The flow request set, $\boldsymbol{R}=\{r_k\}_k$.
b_k	The traffic requirement of r_k.
$\boldsymbol{SC}_k$	The service function chain required by r_k.
$\boldsymbol{SC}_k'$	The service function chain without the last VNF for r_k.
$\boldsymbol{F}$	The set of VNF types for all requests, $\boldsymbol{F}=\{f_j\}_j$.
Variable	**Description**
$\boldsymbol{N}_u$	All adjacent nodes of u.
$w_{u,v}^k$	A binary variable to indicate whether a link between VNF f_j to f_{j+1} for r_k is added to the candidate link set.
$z_{f_j,u}^k$	A binary variable to indicate whether virtual node u is used to represent VNF f_j for r_k.
$\phi_{u,v}$	A auxiliary binary variable to indicate whether a virtual link between virtual node u and v acts as a candidate virtual link.
$m_{f_j,u,v}^k$	A binary variable to indicate whether $\phi_{u,v}$ is involved to construct an acyclic path from VNF f_j to f_{j+1} for r_k.
$\beta_{f_j,u,v}^k$	An auxiliary binary variable to indicate whether $e_{u,v}$ is on the path from VNF f_j to f_{j+1} for r_k.
$\psi_{u,v}$	A binary variable to indicate whether $e_{u,v}$ is included in the virtual graph.
$h_{u,k}^1$	The incoming degree of u in the k^{th} graph.
$g_{u,k}^1$	An auxiliary binary variable to indicate whether the incoming degrees of node u equals to 0 in the k^{th} graph.
$h_{u,k}^2$	The out-going degree of u in the k^{th} graph.
$g_{u,k}^2$	An auxiliary binary variable to indicate whether the out-going degrees of node u equals to 0 in the k^{th} graph.
$g_{u,k}$	An auxiliary binary variable to indicate whether u only has incoming or out-going links in the k^{th} graph.
$D_u^{f_j}$	The processing resource demand of virtual node u holding VNF f_j.
$T_u^{f_j}$	A binary variable to indicate whether virtual node u is used to represent VNF f_j.

iterations. Otherwise, nodes on the loop will still remain and can not be deleted. ■

A. Problem Formulation

In the node-link ILP model $\boldsymbol{P}_s$, the objective function includes two parts. The first part determined by binary variable $T_u^{f_j}$, represents the total number of virtual nodes in the acyclic virtual graph. The second part determined by binary variable $\phi_{u,v}$, denotes the total number of virtual links. When the obtained VNF-FG is transferred to cloud IoT platform for embedding known as VNF-FGE, which is beyond the research scope of this paper, a virtual topology of smaller size can achieve more efficient embedding and resource allocation. Coefficients β_1 and β_2 quantify linear sum of the two. Notations in $\boldsymbol{P}_s$ are summarized in Table I.

$$\boldsymbol{P}_s: \quad \text{Minimize} \ \beta_1 \sum_{u\in\boldsymbol{V}}\sum_{f_j\in\boldsymbol{F}} T_u^{f_j} + \beta_2 \sum_{u,v\in\boldsymbol{V}} \psi_{u,v} \quad (1)$$

subject to:

$$\sum_{u\in\boldsymbol{V}} z_{f_j,u}^k k_u^j = 1, \forall r_k\in\boldsymbol{R}, \forall f_j\in\boldsymbol{SC}_k \quad (2)$$

$$w_{u,v}^k = z_{f_j,u}^k z_{f_{j+1},v}^k, \forall f_j\in\boldsymbol{SC}_k', \forall r_k\in\boldsymbol{R}, \forall u,v\in\boldsymbol{V} \quad (3)$$

$$\frac{z_{f_j,u}^k + z_{f_{j+1},v}^k}{2} \geq w_{u,v}^k \geq z_{f_j,u}^k + z_{f_{j+1},v}^k - 1, \quad \forall f_j\in\boldsymbol{SC}_k', \forall r_k\in\boldsymbol{R}, \forall u,v\in\boldsymbol{V} \quad (4)$$

$$M\sum_{r_k\in\boldsymbol{R}}\sum_{f_j\in\boldsymbol{SC}_k'} w_{u,v}^k \geq \phi_{u,v} \geq \frac{1}{M}\sum_{r_k\in\boldsymbol{R}}\sum_{f_j\in\boldsymbol{SC}_k'} w_{u,v}^k, \forall u,v\in\boldsymbol{V} \quad (5)$$

$$m_{f_j,u,v}^k \leq \phi_{u,v}, \forall f_j\in\boldsymbol{SC}_k', \forall u,v\in\boldsymbol{V}, \forall r_k\in\boldsymbol{R} \quad (6)$$

$$\sum_{v\in\boldsymbol{V}} \phi_{u,v} m_{f_j,u,v}^k - \sum_{v\in\boldsymbol{V}} \phi_{v,u} m_{f_j,v,u}^k = z_{f_j,u}^k - z_{f_{j+1},u}^k, \forall u\in\boldsymbol{V}, \forall r_k\in\boldsymbol{R}, \forall f_j\in\boldsymbol{SC}_k' \quad (7)$$

$$\sum_{v\in\boldsymbol{V}} \beta_{f_j,u,v}^k - \sum_{v\in\boldsymbol{V}} \beta_{f_j,v,u}^k = z_{f_j,u}^k - z_{f_{j+1},u}^k, \quad \forall u\in\boldsymbol{V}, \forall r_k\in\boldsymbol{R}, \forall f_j\in\boldsymbol{SC}_r' \quad (8)$$

$$\beta_{f_j,u,v}^k = \phi_{u,v} m_{f_j,u,v}^k, \forall u,v\in\boldsymbol{V}, \forall r_k\in\boldsymbol{R}, \forall f_j\in\boldsymbol{SC}_r' \quad (9)$$

$$\frac{\phi_{u,v} + m_{f_j,u,v}^r}{2} \geq \beta_{f_j,u,v}^k \geq \phi_{u,v} + m_{f_j,u,v}^k - 1, \quad \forall u,v\in\boldsymbol{V}, \forall r_k\in\boldsymbol{R}, \forall f_j\in\boldsymbol{SC}_r' \quad (10)$$

$$\sum_{r_k\in\boldsymbol{R}}\sum_{f_j\in\boldsymbol{SC}_k'} m_{f_j,u,v}^k \geq \psi_{u,v} \geq \frac{1}{M}\sum_{r_k\in\boldsymbol{R}}\sum_{f_j\in\boldsymbol{SC}_k'} m_{f_j,u,v}^k, \forall u,v\in\boldsymbol{V} \quad (11)$$

$$D_u^{f_j} = \sum_{r_k\in\boldsymbol{R}} z_{f_j,u}^k b_k, \forall u\in\boldsymbol{V}, \forall f_j\in\boldsymbol{F} \quad (12)$$

$$M\sum_{r_k\in\boldsymbol{R}} z_{f_j,u}^k \geq T_u^{f_j} \geq \frac{1}{M}\sum_{r_k\in\boldsymbol{R}} z_{f_j,u}^k, \forall u\in\boldsymbol{V}, \forall f_j\in\boldsymbol{F} \quad (13)$$

$$h_{u,1}^1 = \sum_{v\in\boldsymbol{V}} \psi_{v,u}, \forall u,v\in\boldsymbol{V} \quad (14)$$

$$h_{u,1}^2 = \sum_{v\in\boldsymbol{V}} \psi_{u,v}, \forall u,v\in\boldsymbol{V} \quad (15)$$

$$\frac{1}{M}h_{u,k}^1 \leq 1 - g_{u,k}^1 \leq M h_{u,k}^1, \forall u\in\boldsymbol{V}, k=1,\ldots,|\boldsymbol{V}| \quad (16)$$

$$\frac{1}{M}h_{u,k}^2 \leq 1 - g_{u,k}^2 \leq M h_{u,k}^2, \forall u\in\boldsymbol{V}, k=1,\ldots,|\boldsymbol{V}| \quad (17)$$

$$g_{u,k} = g_{u,k}^1 + g_{u,k}^2, \forall u\in\boldsymbol{V}, k=1,\ldots,|\boldsymbol{V}| \quad (18)$$

$$g_{u,1}' = g_{u,1}^1 g_{u,1}^2, \forall u\in\boldsymbol{V} \quad (19)$$

$$\frac{g_{u,1}^1 + g_{u,1}^2}{2} \geq g_{u,1}' \geq g_{u,1}^1 + g_{u,1}^2 - 1, \forall u\in\boldsymbol{V} \quad (20)$$

$$h_{u,k}^1 = \sum_{v\in\boldsymbol{V}} \psi_{v,u}\big(1 - g_{v,k-1}\big), \forall u\in\boldsymbol{V}, k=2,\ldots,|\boldsymbol{V}| \quad (21)$$

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

$$h^2_{u,k} = \sum_{v \in \boldsymbol{V}} \psi_{u,v}\left(1 - g_{v,k-1}\right), \forall u \in \boldsymbol{V}, k = 2, \ldots, |\boldsymbol{V}| \tag{22}$$

$$\sum_{f_j \in \boldsymbol{F}} \sum_{u \in \boldsymbol{V}} T^{f_j}_u - \left(\sum_{u \in \boldsymbol{V}} g_{u,|\boldsymbol{V}|} - \sum_{u \in \boldsymbol{V}} g'_{u,1} \right) = 0 \tag{23}$$

In (2), binary variable $z^k_{f_j,u}$ indicates whether VNF f_j of flow request r_k is represented by virtual node u, and only one virtual node of type f_j can be selected to fulfill this representation. In (3), binary variable $w^k_{u,v}$ demonstrates whether a candidate link between virtual node u and v for r_k should be considered. If and only if f_j and f_{j+1} choose u and v to fulfill representations respectively, $w^k_{u,v}$ equals to 1, otherwise 0. Since two product terms in (3) are binary variables, which makes it nonlinear, we further construct (4) to obtain a linear form of (3). In (5), binary variable $\phi_{u,v}$ aggregates $w^k_{u,v}$ respecting different requests, in order to form a candidate link set. In (6) and (7), the multiplication of binary variable $m^k_{f_j,u,v}$ and $\phi_{u,v}$ constructs a virtual service path for each flow request. Note that, we assume there are no identical VNF types in each SFC. Equation (8)-(10) are used to replace the nonlinear constraint (7) by introducing auxiliary variable $\beta^k_{f_j,u,v}$. In (11), binary variable $\psi_{u,v}$ denotes whether virtual link between u and v is in the yielded virtual graph. In (12), variable $D^{f_j}_u$ denotes the amount of resources required by virtual node u of f_j type. In (13), binary variable $T^{f_j}_u$ indicates whether u is selected to represent f_j. As descried in Lemma 1, (14)-(22) are to repeatedly prune terminal nodes and the connected links based on a virtual graph composed of $\psi_{u,v}$, with $|\boldsymbol{V}|$ iterations totally. Specifically, to obtain incoming and out-going degrees of each virtual node, (14) and (15) are used respectively at the beginning of the iteration, while (21) and (22) are used in other iterations. In (16) and (17), that binary variable $g^1_{u,k}$ $(resp.\ g^2_{u,k})$ equals to 1 means the incoming $(resp.$ out-going) degrees of u is 0 in the k^{th} iteration, otherwise, 1. In (18), auxiliary binary variable $g_{u,k} = 1$ demonstrates u has no incoming or out-going links in the k^{th} iteration. Briefly speaking, in the pruning process, (16)-(18) are used to find out whether a virtual node only has incoming or out-going links in the k^{th} graph to determine whether pruning is needed. Additionally, it is worth noting that at the beginning, there may be one or more nodes that are isolated with neither incoming nor outgoing links. This is represented by $g'_{u,1} = 1$ in (19) and linearized as in (20). After $|\boldsymbol{V}|$ iterations, there should be nodes or links left, as long as the yielded virtual graph is acyclic, which is constrained by (23). Specifically, $\sum_{u \in \boldsymbol{V}} g_{u,|\boldsymbol{V}|} - \sum_{u \in \boldsymbol{V}} g'_{u,1}$ refers to the number of nodes that are not isolated at the beginning of pruning while having no incoming or outgoing links after $|\boldsymbol{V}|$ times of pruning.

The main idea of $\boldsymbol{P}_s$ programming is that, we first make a candidate virtual node set $\boldsymbol{V}$ as input, each element $u \in \boldsymbol{V}$ is an independent virtual node corresponding to a specific VNF (e.g., f_j) in SFC of individual flows (e.g., r_k). Based on $\boldsymbol{V}$, a forwarding path is pursued for each flow. Afterwards, terminal node pruning is performed repeatedly to guarantee the achieved virtual graph is acyclic. Decision variables $z^k_{f_j}$ and $m^k_{f_j,u,v}$ determine the necessary virtual nodes and links for r_k. An acyclic virtual graph is constructed by auxiliary variables $T^{f_j}_u$ and $\psi_{u,v}$. As the problem of VNF-FG design is NP-hard, we further design an efficient approximation algorithm ensued.

Algorithm 1: Acyclic Forwarding Graph Construction (AFGC) Algorithm

Input: $\boldsymbol{G}_{in} = (\boldsymbol{V}_{in}, \boldsymbol{E}_{in})$
Output: $\boldsymbol{G}_{out} = (\boldsymbol{V}_{out}, \boldsymbol{E}_{out})$

```
while true do
    find all loops in G_in to obtain loop set C;
    sort C in descending order, c_i ∈ C;
    let k = 0, m = 0;
    while i ≤ |C| do
        G^i_in ← G_in;
        run Algorithm 2;
        G_out ← G^i_out;
        j ← i;
        while j ≤ |C| − 1 do
            if c_{j+1} is fully overlapped by c_j then
                j++;
                k++;
        i ← j + 1;
        m ← m + k;
    if m = 0 then
        break;
    let G_in = G_out;
return G_out;
```

B. Algorithm Design

According to ***loop elimination procedure***, direct consolidation for SFCs is performed at first to obtain a direct consolidation graph $\boldsymbol{G}_{in}$, based on which DFS is leveraged to find all loops. Then, we design AFGC algorithm illustrated in Algorithm 1 emphasizing target loop selection, where sub-algorithm LE is further developed in Algorithm 2 to eliminate target loops individually.

Lemma 2: Given $\boldsymbol{SC}_k$ for every flow request $r_k \in \boldsymbol{R}$, the competitive ratio of algorithm AFGC is $O(1 + (|\boldsymbol{R}| - 1)\sigma), \sigma \in (0, 1)$.

Proof: Let C_AFGC and $C\_\boldsymbol{P}_s$ be sum of total number of virtual nodes and links based on AFGC and $\boldsymbol{P}_s$ separately. We define the competitive ratio as $\frac{C\_AFGC}{C\_\boldsymbol{P}_s} = \frac{C\_M + C\_\boldsymbol{P}_s}{C\_\boldsymbol{P}_s} = 1 + \frac{C\_M}{C\_\boldsymbol{P}_s}$, where C_M is the redundant nodes and links generated by AFGC compared to $\boldsymbol{P}_s$. H_k denotes the total number of virtual nodes and links originally required by r_k. Based on AFGC, virtual nodes and links are largely integrated as long as no loop appears. Thus, $C\_M \leq \sum_{k=1}^{|\boldsymbol{R}|-1} H_k \leq (|\boldsymbol{R}| - 1) \times max\{H_k\}_k$, where $max\{H_k\}_k$ denotes the maximal $H_k, \forall r_k \in \boldsymbol{R}$. Then, $\frac{C\_M}{C\_\boldsymbol{P}_s} \leq \frac{(|\boldsymbol{R}|-1) \times max\{H_k\}_k}{C\_\boldsymbol{P}_s} = (|\boldsymbol{R}| - 1) \times \sigma$. Since $max\{H_k\}_k$ can be seen as the total number of

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Loop Elimination (LE) Algorithm

Input: $\boldsymbol{G}^i_{in} = (\boldsymbol{V}^i_{in}, \boldsymbol{E}^i_{in})$, c_i, $\boldsymbol{P}_i = \varnothing$, $\boldsymbol{P}^t_i = \varnothing$
Output: $\boldsymbol{G}^i_{out} = (\boldsymbol{V}^i_{out}, \boldsymbol{E}^i_{out})$

find every flow $f_{ij} \in \boldsymbol{FL}_i$ and its corresponding path p_{ij} on the circle c_i;
Sort $\boldsymbol{FL}_i$ regarding the hops hop_{ij} of path p_{ij};
Find the flow (or flows) in $\boldsymbol{FL}_i$ with maximum hops and put its path into set $\boldsymbol{P}_i$;
for $p_{ij} \in \boldsymbol{P}_i$ **do**
    let m_{ij} = the number of flows on the loop and traversing the last hop of p_{ij};
let $p^t = \underset{p_{ij} \in \boldsymbol{P}_i}{argmin}\ m_{ij}$;
for $p_{ij} \in \boldsymbol{P}_i$ **do**
    if $m_{ij} = p^t$ **then**
        $\boldsymbol{P}^t_i \leftarrow p_{ij}$
for $k = |\boldsymbol{P}^t_i|, \ldots, 1$ **do**
    let $h_k = 0$;
    $\boldsymbol{G}^i_{out,k} \leftarrow \boldsymbol{G}^i_{in}$;
    for $l = |p_k|, \ldots, 1$ **do**
        let VNF_l and VNF_{l+1} denote the head and tail VNFs of the l^{th} virtual link on p_k;
        $h_{k,l} \leftarrow$ the number of flows on the loop and passing through VNF_{l+1};
        $\boldsymbol{G}^i_{out,k} \leftarrow$ increase the number of instances of VNF_{l+1} in $\boldsymbol{G}^i_{out,k}$ to $h_{k,l}$, make p_k monopolize one instance;
        $h_k \leftarrow h_k + h_{k,l}$;
        let n_l = the number of flows on the loop and passing through the l^{th} virtual link;
        if $n_l \leq 1$ **then**
            break
    $\boldsymbol{G}^i_{out} = \underset{\boldsymbol{G}^i_{out,k}}{argmin}\ h_k$;
return $\boldsymbol{G}^i_{out}$;

nodes and links required by a single request which is less than $C\_\boldsymbol{P}_s$, then, $\sigma \in (0, 1)$. Thus, $\frac{C\_AFGC}{C\_\boldsymbol{P}_s} \leq 1 + (|\boldsymbol{R}| - 1)\sigma$.

V. VNF-FG Design in Dynamic Scenario

In dynamic scenario, VNF-FG augmentation is investigated respecting an SFC on the fly to enable resource efficiency and service stability. $\boldsymbol{P}_d$ is first formulated giving importance to loop prevention and reuse of pre-exist virtual topology. Then, approximation algorithm DFGU with a competitive ratio to $\boldsymbol{P}_d$ is designed to achieve VNF-FG extention. When a device is disconnected from the IoT terminal, we can directly remove the virtual nodes and links monopolized by it from the VNF-FG.

A. Problem Formulation

In node-link ILP model $\boldsymbol{P}_d$, a flow request r on the fly joins the IoT terminal. The objective function includes two parts. The first part $\sum_{f_j \in \boldsymbol{SC}_r} \phi^r_{f_j}$ indicates the number of new instances needed to serve r. The second part $\sum_{u,v \in \boldsymbol{V}'} \phi^{new}_{u,v}$ indicates the number of new virtual links required. Coefficients λ_1 and λ_2 quantify linear sum of the two. Notations in $\boldsymbol{P}_d$ are summarized in Table II.

TABLE II
Notation for Problem $\boldsymbol{P}_d$

Parameter	Description
$\boldsymbol{G}$	Current virtual graph $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$.
$\boldsymbol{V}$	The set of virtual nodes, $\boldsymbol{V} = \{u\}$.
$\boldsymbol{E}$	The set of virtual links, $\boldsymbol{E} = \{e_{u,v} : u, v \in \boldsymbol{V}\}$.
$\phi_{u,v}$	A binary parameter to indicate whether $e_{u,v}$ is included in $\boldsymbol{G}$.
r	The new arriving service request.
$\boldsymbol{SC}_r$	The SFC of r, $\boldsymbol{SC}_r = \{f^r_j\}_j$.
$\boldsymbol{SC}'_r$	The SFC without the last VNF for r.
$v^r_{f_j}$	Newly added feasible virtual node holding $f^r_j \in \boldsymbol{SC}_r$.
$\boldsymbol{V}^r$	Newly added feasible virtual node set, $\boldsymbol{V}^r = \{v^r_{f_j}\}$.
$\boldsymbol{F}$	VNF type set in $\boldsymbol{G}$, $\boldsymbol{F} = \{f_j\}_j$.
$\boldsymbol{V}^{\boldsymbol{F}}_{f_j}$	Virtual node set of f_j in $\boldsymbol{G}$, $f_j \in \boldsymbol{F}$, $\boldsymbol{V} = \{\boldsymbol{V}^{\boldsymbol{F}}_{f_j} : f_j \in \boldsymbol{F}\}$.
$\boldsymbol{F}'$	Augmented VNF type set, $\boldsymbol{F}' = \boldsymbol{SC}_r \cup \boldsymbol{F}$.
$\boldsymbol{V}_{f_j}$	The available virtual node set for f_j, $f_j \in \boldsymbol{F}'$.
$\boldsymbol{V}'$	The available virtual node set, $\boldsymbol{V}' = \boldsymbol{V} \cup \boldsymbol{V}^r = \{\boldsymbol{V}_{f_j} : f_j \in \boldsymbol{F}'\}$.
Variable	**Description**
$w^r_{u,v}$	A binary variable to indicate whether link $e_{u,v}$ is newly added to connect VNF f_j to f_{j+1} for r.
$z^r_{f_j,u}$	A binary variable to indicate whether VNF f_j of r is placed on virtual node u.
$\phi'_{u,v}$	A binary variable to indicate whether $e_{u,v}$ is sufficient but may be not necessary in the yielded graph after serving r.
$\phi_{u,v}$	A binary variable to indicate whether link $e_{u,v}$ is involved in the SFC path for r.
$m^r_{f_j,u,v}$	A binary variable to indicate whether $\phi'_{u,v}$ is involved when finding acyclic virtual path from VNF f_j to f_{j+1} for r.
$\lambda^r_{f_j,u,v}$	An auxiliary binary variable to represent the multiplication of $m^r_{f_j,u,v}$ and $\phi'_{u,v}$.
$\psi_{u,v}$	An auxiliary binary variable to indicate whether $e_{u,v}$ is involved in the yielded virtual graph $\boldsymbol{G}_{out}$.
$h^1_{u,k}$	Incoming degrees of u in residual topology $\boldsymbol{G}^k_{out}$ after k iterations of deletion operation on $\boldsymbol{G}_{out}$.
$g^1_{u,k}$	A binary variable to indicate whether incoming degrees of node u equals to 0 in $\boldsymbol{G}^k_{out}$.
$h^2_{u,k}$	Out-going degrees of u in residual topology $\boldsymbol{G}^k_{out}$ after k iterations of deletion operation on $\boldsymbol{G}_{out}$.
$g^2_{u,k}$	A binary variable to indicate whether out-going degrees of node u equals to 0 in $\boldsymbol{G}^k_{out}$.
$g_{u,k}$	A binary variable to indicate whether u only has incoming or out-going links in $\boldsymbol{G}^k_{out}$.
$y^r_{f_j}$	A binary variable to indicate whether a virtual node of f_j is newly added into $\boldsymbol{G}_{out}$ to serve r.
$\phi^{new}_{u,v}$	A binary variable to indicate whether $e_{u,v}$ is newly added into $\boldsymbol{G}_{out}$ to serve r.

$$\boldsymbol{P}_d:\ \text{Minimize}\quad \lambda_1 \sum_{f_j \in \boldsymbol{SC}_r} y^r_{f_j} + \lambda_2 \sum_{u,v \in \boldsymbol{V}'} \phi^{new}_{u,v} \tag{24}$$

$$\text{subject to:}\quad \boldsymbol{V}^1_{f_j} = \boldsymbol{V}^{\boldsymbol{F}}_{f_j} \cup v^r_{f_j}, \forall f^r_j \in \boldsymbol{F} \tag{25}$$

$$\boldsymbol{V}^2_{f_j} = v^r_{f_j}, \forall f^r_j \notin \boldsymbol{F} \tag{26}$$

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

$$\boldsymbol{V}_{f_j} = \boldsymbol{V}_{f_j}^1 \cup \boldsymbol{V}_{f_j}^2, \forall f_j \in \boldsymbol{F}' \tag{27}$$

$$\sum_{u \in \boldsymbol{V}_{f_j}} z_{f_j,u}^r = 1, \forall f_j \in \boldsymbol{SC}_r \tag{28}$$

$$\sum_{u \in \boldsymbol{V}'} z_{f_j,u}^r = 1, \forall f_i \in \boldsymbol{SC}_r \tag{29}$$

$$w_{u,v}^r = z_{f_j,u}^r z_{f_{j+1},v}^r \big(1 - \phi_{u,v}\big), \forall f_j \in \boldsymbol{SC}'_r, \forall u, v \in \boldsymbol{V}' \tag{30}$$

$$\frac{z_{f_j,u}^r + z_{f_{j+1},v}^r + \big(1 - \phi_{u,v}\big)}{3} \geq w_{u,v}^r \geq z_{f_j,u}^r + z_{f_{j+1},v}^r + \big(1 - \phi_{u,v}\big) - 2, \forall f_j \in \boldsymbol{SC}'_r, \forall u, v \in \boldsymbol{V}' \tag{31}$$

$$\phi'_{u,v} = \phi_{u,v} + w_{u,v}^r, \forall u, v \in \boldsymbol{V}' \tag{32}$$

$$m_{f_j,u,v}^r \leq \phi'_{u,v}, \forall f_j \in \boldsymbol{SC}'_r, \forall u, v \in \boldsymbol{V}' \tag{33}$$

$$\sum_{u,v \in \boldsymbol{V}} \phi'_{u,v} m_{f_j,u,v}^r - \sum_{u,v \in \boldsymbol{V}} \phi'_{v,u} m_{f_j,v,u}^r = z_{f_j,u}^r - z_{f_{j+1},u}^r, \forall u \in \boldsymbol{V}', \forall f_j \in \boldsymbol{SC}'_r \tag{34}$$

$$\sum_{u,v \in \boldsymbol{V}} \lambda_{f_j,u,v}^r - \sum_{u,v \in \boldsymbol{V}} \lambda_{f_j,v,u}^r = z_{f_j,u}^r - z_{f_{j+1},u}^r, \forall u \in \boldsymbol{V}', \forall f_j \in \boldsymbol{SC}'_r \tag{35}$$

$$\lambda_{f_j,u,v}^r = \phi'_{u,v} m_{f_j,u,v}^r, \forall u, v \in \boldsymbol{V}', \forall f_j \in \boldsymbol{SC}'_r \tag{36}$$

$$\frac{\phi'_{u,v} + m_{f_j,u,v}^r}{2} \geq \lambda_{f_j,u,v}^r \geq \phi'_{u,v} + m_{f_j,u,v}^r - 1, \forall u, v \in \boldsymbol{V}', \forall f_j \in \boldsymbol{SC}'_r \tag{37}$$

$$\psi_{u,v} = \sum_{f_j \in \boldsymbol{SC}'_r} m_{f_j,u,v}^r + \phi_{u,v}, \forall u, v \in \boldsymbol{V}' \tag{38}$$

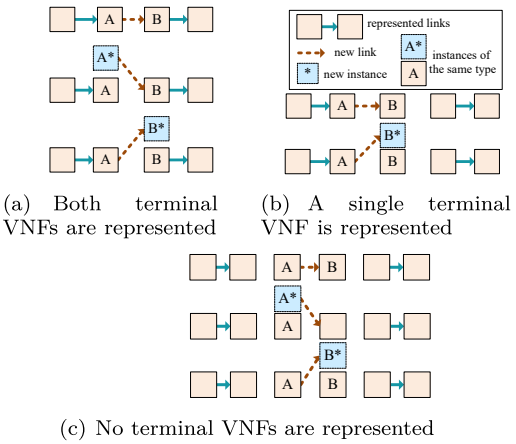
$$\sum_{f_j \in \boldsymbol{SC}_r} y_{f_j}^r + |\boldsymbol{V}| - \left(\sum_{u \in \boldsymbol{V}'} g_{u,|\boldsymbol{V}|} - \sum_{u \in \boldsymbol{V}'} g'_{u,1} \right) = 0 \tag{39}$$

$$y_{f_j}^r = \sum_{u \in \boldsymbol{V}^r} z_{f_j,u}^r, \forall f_j \in \boldsymbol{SC}_r \tag{40}$$

$$\phi_{u,v}^{new} = \sum_{f_j \in \boldsymbol{SC}'_r} m_{f_j,u,v}^r \big(1 - \phi_{u,v}\big), \forall u, v \in \boldsymbol{V}' \tag{41}$$

while under (14) to (22) by substituting $|\boldsymbol{V}|$ by $|\boldsymbol{V}'|$.

In (25), $\boldsymbol{V}_{f_j}^1$ denotes feasible virtual node set of VNF type f_j^r required by r, which already exists in VNF type set $\boldsymbol{F}$ of current graph $\boldsymbol{G}$. In (26), $\boldsymbol{V}_{f_j}^2$ denotes newly added virtual node of f_j^r that is different from any type in $\boldsymbol{F}$. Equation (27) is to integrate the two sets for the ease of expression. Note that, $\boldsymbol{V}_{f_j}$ can also be decomposed into $\boldsymbol{V}_{f_j}^{\boldsymbol{F}}$ and $v_{f_j}^r$. Equation (28) denotes that, each VNF $f_j \in \boldsymbol{SC}_r$ should be represented either by a virtual node in current virtual node set $\boldsymbol{V}_{f_j}^{\boldsymbol{F}}$, or by newly added virtual node $v_{f_j}^r$. This guarantees that in the worst case an acyclic path of r can be constructed by adding a new virtual node of f_j. Equation (29) constraints every VNF of r can only be represented by a single virtual node. In (30), binary variable $w_{u,v}^r$ equals to 1 if and only if two adjacent VNFs of r are represented by virtual node u and v separately while no links between the two nodes existing in $\boldsymbol{G}$. Considering that (30) is not linear as the two product terms $z_{f_j,u}^r$ and $z_{f_{j+1},u}^r$ are decision variables, we further replace it with the linear form (31). In (32), binary variable $\phi'_{u,v}$ is the sum of current links denoted by $\phi_{u,v}$ that already exist in $\boldsymbol{G}$, and the candidate links additionally constructed for r, in order to indicate whether $e_{u,v}, \forall u, v \in \boldsymbol{V}'$ exists in the candidate graph. In (33) and (34), a binary variable $m_{f_j,u,v}^r$ is introduced, based on which a virtual service path of r is obtained. Equation (35)-(37) are used to deal with nonlinear constraint (34) by introducing another auxiliary binary variable $\lambda_{f_j,u,v}^r$. In (38), another binary variable $\psi_{u,v}$ is used to denote whether $e_{u,v}$ exists in the augmented virtual graph ultimately yielded. At last, (39) guarantee the resulting augmented virtual graph has no loops by constraining no nodes remain after $|\boldsymbol{V}'|$ delete operations. In (41), binary variable $y_{f_j}^r$ indicates whether a new virtual node of f_j is added to the augmented graph, which equals to the difference between the total number of virtual nodes of f_j in augmented graph and that in current graph $\boldsymbol{G}$. In (41), binary variable $\phi_{u,v}^{new}$ denotes whether a new virtual link $e_{u,v}, \forall u, v \in \boldsymbol{V}'$ is added to the augmented graph. If $e_{u,v}$ only exists in augmented graph rather than $\boldsymbol{G}$, $\phi_{u,v}^{new} = 1$, otherwise, 0.



(a) Both terminal VNFs are represented (b) A single terminal VNF is represented

(c) No terminal VNFs are represented

Fig. 4. The illustration for 3 types of unrepresented links.

B. Algorithm Design

The main idea of approximation algorithm DFGU for VNF-FG augmentation is to classify virtual links of newly arriving SFC and integrate them with current virtual graph sequentially. That a virtual link can be represented, means in current graph there is a path whose head and tail are of identical VNF types as that of it. Classification of a virtual link depends on whether it can be represented by an existing path and whether the VNF types of its head and tail exist in current graph. If all virtual links can be represented, the VNF-FG remains unchanged. Otherwise, unrepresented links are divided into three types with different operations. (i) ***Type* 1**: as shown in Fig. 4(a), the head and tail of an unrepresented link are already represented by node A and B respectively in current graph. It is better to reuse existing nodes and link them by a new virtual link. If a loop appears, then a new virtual node A^* or B^* is initialized. (ii) ***Type* 2**: as in Fig. 4(b), either the head or tail is represented by existing node (e.g., A). Then, if the type of the unrepresented VNF exists in current VNF set, select an appropriate existing node (e.g., B) of right this type, and link A and B if no loop appears. Otherwise, add a new node B^* and link the two likewise. (iii) ***Type* 3**: as in Fig. 4(c), none

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3: Dynamic Forwarding Graph Updating (DFGU) Algorithm

Input: $\boldsymbol{G}_{in} = (\boldsymbol{V}_{in}, \boldsymbol{E}_{in})$, new arriving service request SFC_r
Output: $\boldsymbol{G}_{out} = (\boldsymbol{V}_{out}, \boldsymbol{E}_{out})$

for $i = 1, \ldots, |SFC_r| - 1$ **do**
  let $m_i = 0$;
  if *the ith virtual link* l_i *can be represented by one or several paths in* $\boldsymbol{G}_{in}$ **then**
    choose the minimum hop path, and use its head and tail VNF instances, denoted by $map^{i}_{i,i+1}$ and $map^{i+1}_{i,i+1}$, to hold the *ith* and $(i+1)th$ VNFs of SFC_r;
    let $m_i = 1$;
    if *i>1* **then**
      if $m_{i-1} = m_i$ **then**
        if $map^{i}_{i-1,i} \neq map^{i}_{i,i+1}$ **then**
          let A=$hops(map^{i-1}_{i-1,i}, map^{i}_{i-1,i}) + hops(map^{i}_{i-1,i}, map^{i+1}_{i,i+1})$;
          let B=$hops(map^{i-1}_{i-1,i}, map^{i}_{i,i+1}) + hops(map^{i}_{i,i+1}, map^{i+1}_{i,i+1})$;
          if *A>B* **then**
            $map^{i}_{i-1,i} \leftarrow map^{i}_{i,i+1}$;
          else
            $map^{i}_{i,i+1} \leftarrow map^{i}_{i-1,i}$;

$\boldsymbol{G}'_{in} \leftarrow \boldsymbol{G}_{in}$;
for $i = 1, \ldots, |SFC_r| - 1$ **do**
  if $m_i = 0$ **then**
    if *ith and* $(i+1)th$ *VNFs in* l_i *are represented* **then**
      $Li \leftarrow$ a new virtual link from $map^{i}_{i,i+1}$ to $map^{i+1}_{i,i+1}$;
      $\boldsymbol{G}'_{in} \leftarrow \boldsymbol{G}'_{in} \cup Li$;
      if $\boldsymbol{G}'_{in}$ *is cyclic* **then**
        add a new instance INS_i (or INS_{i+1}) for *ith* VNF (or $(i+1)th$);
        $map^{i}_{i,i+1} \leftarrow INS_i$ (or $map^{i+1}_{i,i+1} \leftarrow INS_{i+1}$);
        $Li^* \leftarrow$ a new virtual link from $map^{i}_{i,i+1}$ to $map^{i+1}_{i,i+1}$;
        $\boldsymbol{G}'_{in} \leftarrow \boldsymbol{G}'_{in} \cup Li^* \cup INS_i (or INS_{i+1})$;
    else
      if *only the head (or tail) of* l_i *is represented* **then**
        run Algorithm 4;
      else
        run Algorithm 5;
      $\boldsymbol{G}'_{in} \leftarrow \boldsymbol{G}^*_{in}$;

$\boldsymbol{G}_{out} \leftarrow \boldsymbol{G}'_{in}$;
return $\boldsymbol{G}_{out}$;

of the head or tail is represented. Existing nodes of identical type as the head or tail have privilege to be reused unless a loop appears.

In the light of above analysis, DFGU algorithm shown in Algorithm 3 and its sub-algorithms Algorithm 4 and Algorithm 5, are well designed to perform VNF-FG augmentation elaborately.

Algorithm 4: Link Consolidation With Single Represented Endpoint

Input: $\boldsymbol{G}'_{in}$, a virtual link l with one endpoint (the head or the tail VNF) represented in $\boldsymbol{G}'_{in}$
Output: $\boldsymbol{G}^*_{in}$

if *the only unrepresented endpoint (e.g.,* VNF_k*) of* l *has existing instances in* $\boldsymbol{G}_{in}$ **then**
  sort the instances and put them in a set $\boldsymbol{S}_k$;
  for $j = 1, \ldots, |\boldsymbol{S}_k| + 1$ **do**
    construct a new link l^j connecting instance of represented endpoint to the *jth* instance of VNF_k, namely INS^i_k;
    $\boldsymbol{G}'_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^j$;
    if *no circle exists in* $\boldsymbol{G}'_{in}$ **then**
      $a \leftarrow j$;
      $l^* \leftarrow l^j$;
      break;
  if $a = |\boldsymbol{S}_k| + 1$ **then**
    construct a new instance $INS^{|\boldsymbol{S}_k|+1}_k$ for VNF_k;
    $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS^{|\boldsymbol{S}_k|+1}_k$;
  else
    VNF_k of l is represented by the already exist INS^a_k in $\boldsymbol{G}'_{in}$;
    $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^*$;
else
  construct a new instance, $INS^b_k, b = 1$, for VNF_k, and connect it to the already represented instance by a new link l^*;
  $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS^b_k$;
return $\boldsymbol{G}^*_{in}$;

Lemma 3: Given $\boldsymbol{SC}_r$ of the incoming flow request *r*, the competitive ratio of algorithm DFGU is $O(1 + \frac{|\boldsymbol{SC}_r|}{|\boldsymbol{F}_r|})$, when the number of new types of VNFs $|\boldsymbol{F}_r| \neq 0$.

Proof: Let C_DFGU and $C\_\boldsymbol{P}_d$ be sum of total number of new virtual nodes and links based on DFGU and $\boldsymbol{P}_d$ separately. We define competitive ratio as $\frac{C\_DFGU}{C\_\boldsymbol{P}_d} = \frac{C\_N + C\_\boldsymbol{P}_d}{C\_\boldsymbol{P}_d} = 1 + \frac{C\_N}{C\_\boldsymbol{P}_d}$. C_N is redundant nodes and links generated by DFGU compared with $\boldsymbol{P}_d$. H_r denotes total number of virtual nodes and links originally required by *r*. $H_r = 2|\boldsymbol{SC}_r| - 1$ that consists of $|\boldsymbol{SC}_r|$ nodes and $|\boldsymbol{SC}_r| - 1$ links. Based on DFGU, existing nodes and links are largely reused by *r* as long as no loop appears. Thus, $C\_N \leq H_r$. On the other hand, if a new VNF type exists in $\boldsymbol{SC}_r$, a new node should be initiated and added to current graph with at least one new link, which makes $C\_\boldsymbol{P}_d \geq 2|\boldsymbol{F}_r|$. As $|\boldsymbol{F}_r| \leq |\boldsymbol{SC}_r|$, $\frac{C\_DFGU}{C\_\boldsymbol{P}_d} = 1 + \frac{C\_N}{C\_\boldsymbol{P}_d} \leq 1 + \frac{2|\boldsymbol{SC}_r|-1}{2|\boldsymbol{F}_r|} \leq 1 + \frac{|\boldsymbol{SC}_r|}{|\boldsymbol{F}_r|}$, when $|\boldsymbol{F}_r| \neq 0$.

C. Complexity Analysis

In AFGC, we first run DFS to find all loops in time complexity of $O(|\boldsymbol{V}|)$. Algorithm 2 is to eliminate a

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

Algorithm 5: Link Consolidation With None Represented Endpoints

Input: $\boldsymbol{G}'_{in}$, a virtual link l with no endpoint (the head or the tail VNF) represented in $\boldsymbol{G}'_{in}$
Output: $\boldsymbol{G}^*_{in}$

if *instances of the head and tail of l (e.g., VNF_m and VNF_n) both exist in* $\boldsymbol{G}'_{in}$ **then**
  sort instances of the two by resource requirement in ascending order;
  put them in set $\boldsymbol{S}_m$ and $\boldsymbol{S}_n$ separately;
  for $i = 1, \ldots, |\boldsymbol{S}_m| + 1$ **do**
    for $j = 1, \ldots, |\boldsymbol{S}_n| + 1$ **do**
      construct a new virtual link l^{ij} from INS_m^i to INS_n^j;
      $\boldsymbol{G}'_{in} = \boldsymbol{G}'_{in} \cup l'$;
      if *no circle exists in* $\boldsymbol{G}'_{in}$ **then**
        $a \leftarrow i, b \leftarrow j$;
        $l^* \leftarrow l^{ij}$;
        break;
  $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^*$;
  if $a = |\boldsymbol{S}_m| + 1$ **then**
    construct a new instance $INS_m^{|\boldsymbol{S}_m|+1}$ of VNF_m;
    $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS_m^{|\boldsymbol{S}_m|+1}$;
    if $b = |\boldsymbol{S}_n| + 1$ **then**
      construct a new instance $INS_n^{|\boldsymbol{S}_n|+1}$ of VNF_n;
      $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS_m^{|\boldsymbol{S}_m|+1} \cup INS_n^{|\boldsymbol{S}_n|+1}$;
  else
    if $b = |\boldsymbol{S}_n| + 1$ **then**
      construct a new instance $INS_n^{|\boldsymbol{S}_n|+1}$ of VNF_n;
      $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS_n^{|\boldsymbol{S}_n|+1}$;
else
  if *only one endpoint of l has existing instances* **then**
    find the instance with least resource requirement;
    create an instance INS_g^1 for the new emerging VNF (e.g., VNF_g) of l ;
    link the two instances by adding a new link l^* in $\boldsymbol{G}_{in}$;
    $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS_g^1$;
  else
    create two instance INS_h^1 and INS_q^1 for each of two new emerging VNFs (e.g., VNF_h, VNF_q) in l;
    link them by adding a new link l^* in $\boldsymbol{G}_{in}$;
    $\boldsymbol{G}^*_{in} \leftarrow \boldsymbol{G}'_{in} \cup l^* \cup INS_h^1 \cup INS_q^1$;
return $\boldsymbol{G}^*_{in}$;

single loop in $O(|\boldsymbol{R}||\boldsymbol{SC}_m|)$, where $|\boldsymbol{SC}_m|$ indicates maximum length of all function chains. Then, Algorithm 1 is to iteratively break loops to build an acyclic virtual graph in $O(|\boldsymbol{V}| + |\boldsymbol{R}||\boldsymbol{SC}_m|)$. In DFGU, three types of unrepresented links are processed individually. Time complexity for *type* 1 takes $O(|\boldsymbol{SC}_r|)$ in Algorithm 3. For *type* 2, Algorithm 4 runs in $O(|\boldsymbol{V}'||\boldsymbol{SC}_r|)$ where DFS is carried out to identify whether a loop appears. Similarly, complexity of Algorithm 5 for *type* 3 is $O(|\boldsymbol{V}'|^2|\boldsymbol{SC}_r|)$. Therefrom, time complexity of DFGU algorithm involving Algorithm 3–5 is $O(|\boldsymbol{V}'|^2|\boldsymbol{SC}_r|)$.

VI. Simulation and Evaluation

To evaluate performances of AFGC and DFGU, extensive simulations are conducted. As few work focuses on the research area of VNF-FG design, we make different modifications to the Flow Design and Combination (FDC) algorithm developed in [24] to obtain FDC-S and FDC-D, which accommodate static and dynamic scenarios respectively. Specifically, FDC-S treats each VNF traversed by multiple flows as the key instance and makes the number of replicas equal to the number of flows passing through. FDC-D is slightly modified based on FDC-S. If a VNF type in newly arriving SFC exists in current VNF-FG, it is directly merged into current VNF-FG. Otherwise, initialize a new instance. Thereafter, FDC-S takes further action. Assessment of $\boldsymbol{P}_s$, AFGC and FDC-S are obtained for static scenario. In dynamic scenario, $\boldsymbol{P}_d$ and DFGU are contrasted together with FDC-D. Generally, the quantity of flow requests, length of SFCs, and proportion of new VNF types in each SFC can affect virtual graph configuration. We change the values of the three factors to explore the performance of different algorithms. All simulation experiments are implemented in an Intel E5-2620 sever with six CPUs and 16GB memory.

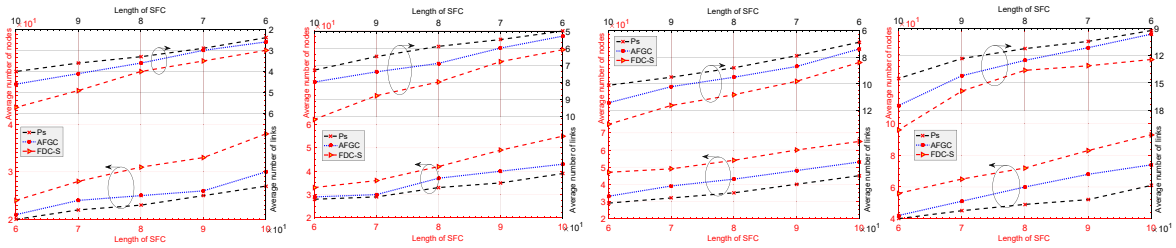
A. Simulation Setup

Distinct VNFs in every SFC are randomly sampled. β_1 and β_2 in $\boldsymbol{P}_s$, and λ_1 and λ_2 in $\boldsymbol{P}_d$ are simply set to 1. We do comparisons over different algorithms in terms of the number of virtual nodes and links in the yielded VNF-FG to obtain fine grained results. Each result is averaged on 50 samples and described as average metrics.

B. Results in Static Scenario

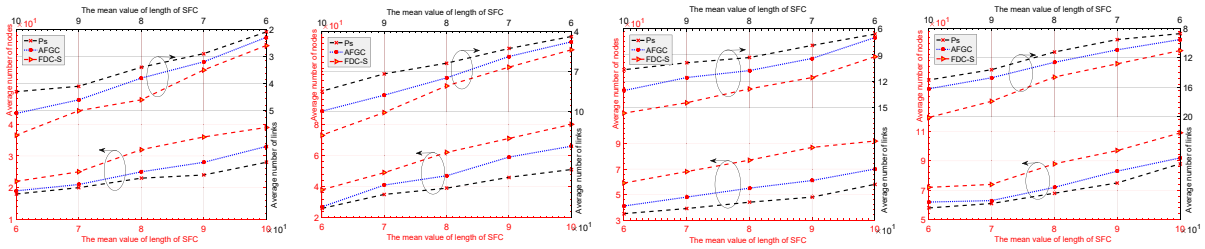
A number of 5, 10, 15, 20 flow requests are injected separately. The length of SFCs are obtained in two ways. One is to make every SFC have identical length (e.g., 6, 7, 8, 9, 10) with the results depicted in Fig. 5. The other is to generate the length of SFC in normal distribution with the results illustrated in Fig. 6. It makes sense that $\boldsymbol{P}_s$ results in the least number of virtual nodes and links. The performance of AFGC is better than FDC-S. This is because, FDC-S simply regards instances traversed by multiple flows as key instances, and requires the number of replicas equal to the number of flows passing through. Contrarily, AFGC runs a thorough loop check, then, begins loop elimination with the biggest one as this may possibly remove the smaller ones inside simultaneously to upgrade efficiency. The key instances are determined based on fine-grained inspection of loop paths, which results in a smaller number of key instances.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.



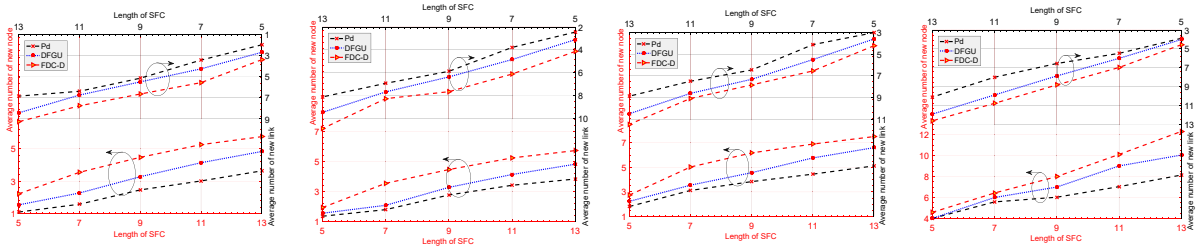
(a) 5 flow requests (b) 10 flow requests (c) 15 flow requests (d) 20 flow requests

Fig. 5. Number of virtual nodes and links in the yielded topology with different flow requests & identical length of SFC.



(a) 5 flow requests (b) 10 flow requests (c) 15 flow requests (d) 20 flow requests

Fig. 6. Number of virtual nodes and links in the yielded topology with different flow requests & length of SFC in normal distribution.



(a) no new VNF type (b) 20% new VNF types (c) 40% new VNF types (d) 80% new VNF types

Fig. 7. Number of additional virtual nodes and links in topology 1 with different ratio of new VNF types.

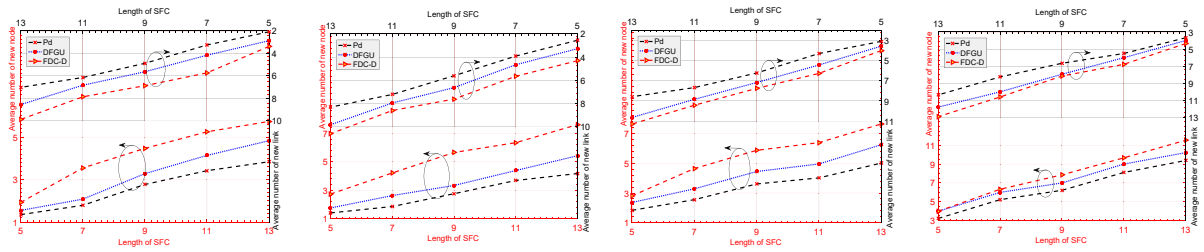
Moreover, relative numerical results of virtual nodes ($resp.$ links) are presented which are defined as $\frac{nodes\_AFGC - nodes\_\boldsymbol{P}_s}{nodes\_\boldsymbol{P}_s}$ ($resp.\frac{links\_AFGC - links\_\boldsymbol{P}_s}{links\_\boldsymbol{P}_s}$), where $nodes\_AFGC$ ($resp.links\_AFGC$) and $nodes\_\boldsymbol{P}_s$ ($resp.links\_\boldsymbol{P}_s$) denote number of virtual nodes ($resp.$ links) in VNF-FG generated by AFGC and $\boldsymbol{P}_s$ separately. Considering 5, 10, 15, 20 requests, AFGC leads to 9.45%, 12.28%, 17.77%, 18.57% increase on the average of different length of SFC. Changing length of SFC to 6, 7, 8, 9, 10, AFGC leads to 12.01%, 13.97%, 14.39%, 14.79%, 15.12% increase on the average of different number of flow requests. Obviously, the gap between AFGC and $\boldsymbol{P}_s$ expands a bit within a reasonable range.

C. Results in Dynamic Scenario

Three artificial topologies, containing 20, 40, 60 nodes and 33, 72, 111 links respectively, are considered and act as background VNG-FG to aggregate a single SFC on the fly. We set length of SFCs to 5, 7, 9, 11, 13, and change ratio of new VNF types to 0%, 20%, 40%, 80%.

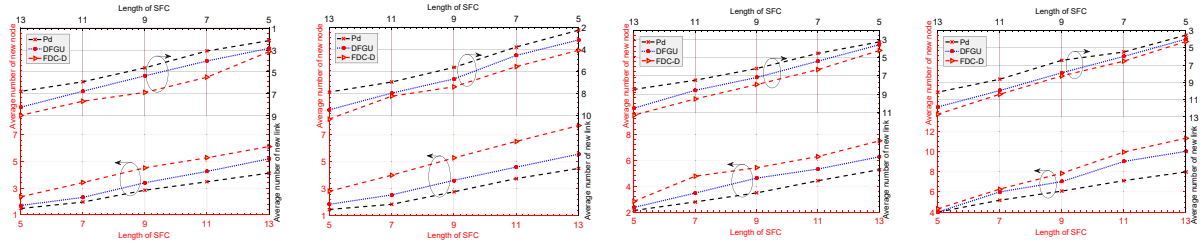
Results shown in Fig. 7–9 indicates that, average number of new virtual nodes and links of DFGU are lower than that of FDC-D. $\boldsymbol{P}_d$ performs best requiring minimum number of new virtual nodes and links. This stems from the fact, DFGU investigates virtual links of the arriving SFC in an orderly fashion and greatly reuses nodes and links of current virtual graph as long as no loop appears, to construct an augmented graph of minimum size. Relative results are defined similar to that of AFGC. When no new VNF types exist in the SFC, at the length of 5,7,9,11,13, DFGU leads to 14.81%, 16.20%, 18.41%, 21.05%, 25.78% increase in topo 1, 15.38%, 16.84%, 19.01%, 21.87%, 24.73% increase in topo 2, and 15.54%, 17.86%, 19.65%, 22.41%, 25.90% increase in topo 3. With the increase of length of SFC, the gap between DFGU and $\boldsymbol{P}_d$ grows greater since it gets harder for DFGU to obtain a DAG while privileging the reuse. When ratios of new VNF types are 20%, 40%, 80%, DFGU leads to an average increase of 28.46%, 23.54%, 15.10% respecting different SFC lengths in topo 1, 27.19%, 22.93%, 14.11% in topo 2, and 28.07%, 23.46%, 15.52% in topo 3. With the increase of the ratios, the distances between DFGU and $\boldsymbol{P}_d$ decrease. This actually verifies the effectiveness of DFGU from another aspect. When fewer VNFs need to consider for reuse, performance of DFGU is closer to that of $\boldsymbol{P}_d$.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.



(a) no new VNF type (b) 20% new VNF types (c) 40% new VNF types (d) 80% new VNF types

Fig. 8. Number of additional virtual nodes and links in topology 2 with different ratio of new VNF types.



(a) no new VNF type (b) 20% new VNF types (c) 40% new VNF types (d) 80% new VNF types

Fig. 9. Number of additional virtual nodes and links in topology 3 with different ratio of new VNF types.

All algorithms are implemented in Python, $\boldsymbol{P}_s$ and $\boldsymbol{P}_d$ are obtained by Gurobi optimizer. Average execution times are 197, 9.12, 5.46 seconds for $\boldsymbol{P}_s$, AFGC, FDC-S respectively in static scenario, 34, 2.87, 1.96 seconds for $\boldsymbol{P}_d$, DFGU, FDC-D in dynamic scenario. When it comes to larger number of SFCs or big virtual topology, the efficiency advantages of AFGC and DFGU over their respective ILP models are more obvious.

VII. Conclusion

In IoT networks, the problem of VNF-FG design is proved to be NP-hard and thoroughly explored in two scenarios. For static scenario, an ILP formulation $\boldsymbol{P}_s$ is established, in which loop prevention constraints are well set and validated by Lemma 1. Then, approximation algorithm AFGC with proved competitive ratio is designed paying close attention to determination of key instances to run loop break comprehensively. For dynamic scenario, we emphasize the reuse of current virtual graph to obtain an augmented graph with compact size. An ILP model $\boldsymbol{P}_d$ is first formulated. Then, approximation algorithm DFGU with proved competitive ratio is developed, which investigate each virtual link of the incoming SFC sequentially and give priority to reuse of existing virtual nodes and links. Extensive simulation results demonstrate the two proposed algorithms perform better than state-of-art work. With respect to dynamic methods for handling large numbers of incoming flow requests and adaptive resource allocation of the generated VNF-FG, we leave these open research problems in our future work.

References

[1] M. B. Mohamed, O. M. Alofe, M. A. Azad, H. S. Lallie, K. Fatema, and T. Sharif, "A comprehensive survey on secure software-defined network for the Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 1, 2022, Art. no. e4391.

[2] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou, "Complementing IoT services through software defined networking and edge computing: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1761–1804, 3rd Quart., 2020.

[3] E. Fazeldehkordi and T.-M. Grønli, "A survey of security architectures for edge computing-based IoT," *IoT*, vol. 3, no. 3, pp. 332–365, 2022.

[4] N. Y.-R. Douha, M. Bhuyan, S. Kashihara, D. Fall, Y. Taenaka, and Y. Kadobayashi, "A survey on blockchain, SDN and NFV for the smart-home security," *Internet Things*, vol. 20, Nov. 2022, Art. no. 100588.

[5] C. Ren, X. Chen, H. Xiang, Y. Wang, Y. Li, and H. Li, "On efficient delay-aware multisource multicasting in NFV-enabled soft-warized networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3371–3386, Sep. 2022.

[6] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–41, 2019.

[7] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *J. Supercomput.*, vol. 78, no. 2, pp. 1983–2014, 2022.

[8] Y. Zhu, H. Yao, T. Mai, W. He, N. Zhang, and M. Guizani, "Multi-agent reinforcement learning aided service function chain deployment for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15674–15684, Sep. 2022.

[9] R. Lin, L. He, S. Luo, and M. Zukerman, "Energy-aware service function chaining embedding in NFV networks," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1158–1171, Mar./Apr. 2023.

[10] I. Sarrigiannis, A. Antonopoulos, K. Ramantas, M. Efthymiopoulou, L. M. Contreras, and C. Verikoukis, "Cost-aware placement and enhanced lifecycle management of service function chains in a multi-domain 5G architecture," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 5006–5020, Dec. 2022.

[11] C. Ren, S. Bai, Y. Wang, and Y. Li, "Achieving near-optimal traffic engineering using a distributed algorithm in hybrid SDN," *IEEE Access*, vol. 8, pp. 29111–29124, 2020.

[12] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 106984.

[13] C. Ren, H. Li, Y. Li, Y. Wang, H. Xiang, and X. Chen, "On efficient service function chaining in hybrid software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1614–1628, Jun. 2022.

[14] M. A. Khoshkholghi and T. Mahmoodi, "Edge intelligence for service function chain deployment in NFV-enabled networks," *Comput. Netw.*, vol. 219, Dec. 2022, Art. no. 109451.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.

[15] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May/Jun. 2016.
[16] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, 2014, pp. 7–13.
[17] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for vnf forwarding graph embedding," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1318–1331, Dec. 2019.
[18] B. Zhang et al., "A survey of VNF forwarding graph embedding in B5G/6G networks," *Wireless Netw.*, pp. 1–24, Aug. 2021.
[19] C. A. Ouedraogo, S. Medjiah, C. Chassot, K. Drira, and J. Aguilar, "A cost-effective approach for end-to-end QoS management in NFV-enabled IoT platforms," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3885–3903, Mar. 2021.
[20] Y. Xie et al., "Virtualized network function forwarding graph placing in SDN and NFV-enabled IoT networks: A graph neural network assisted deep reinforcement learning method," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 524–537, Mar. 2022.
[21] R. Qiu et al., "Virtual network function deployment algorithm based on graph convolution deep reinforcement learning," *J. Supercomput.*, vol. 79, pp. 6849–6870, Apr. 2023.
[22] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, 2015, pp. 89–97.
[23] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 533–546, Sep. 2016.
[24] J. Cao, Y. Zhang, W. An, X. Chen, J. Sun, and Y. Han, "VNF-FG design and VNF placement for 5G mobile networks," *Sci. China Inf. Sci.*, vol. 60, no. 4, 2017, Art. no. 040302.
[25] O. Houidi, O. Soualah, W. Louati, and D. Zeghlache, "Dynamic VNF forwarding graph extension algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1389–1402, Sep. 2020.
[26] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

Cheng Ren received the B.S., M.S., and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2003, 2006, and 2018, respectively. She is an Associate Professor with Southwest Petroleum University, Chengdu. She is currently doing the Postdoctoral Research. Her research interests include network resource optimization, softwarized networks, and computer vision.

Jiangping Zhang received the B.S. degree in mechanical and electronic engineering from the Chengdu University of Information and Technology, Chengdu, China, in 2021. He is currently a Graduate Student with Southwest Petroleum University. His research interests include network resource optimization and machine learning.

Yu Wang received the B.S. degree from Chongqing University in 2010, the M.S. degree from Xi'an Jiaotong University, China, in 2013, and the Ph.D. degree from Kagawa University, Japan, in 2016. She is currently an Associate Professor with Southwest Petroleum University. Her research interests include robotic endovascular surgery, virtual-reality, and underwater robots.

Yaxin Li received the B.S. degree in automation from Chongqing University, China, in 2010, the M.S. degree from the University of Electronic Science and Technology of China in 2013, and the Ph.D. degree from Kagawa University, Japan, in 2016. He is a Professor with Southwest Petroleum University. His current research interests are biomimetic underwater and biomedical-related robotics.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on May 05,2025 at 19:47:07 UTC from IEEE Xplore. Restrictions apply.