Service Function Chain Graph Transformation for Enhanced Resource Efficiency in NFV

Angelos Pentelas and Panagiotis Papadimitriou

Department of Applied Informatics, University of Macedonia, Greece

{*apentelas, papadimitriou*}*@uom.edu.gr*

***Abstract*—Service Function Chain (SFC) embedding optimization is crucial for the resource efficiency of Network Function Virtualization infrastructures (NFVI). Nevertheless, high utilization and/or fragmentation levels of a NFVI can significantly restrict the feasible solution space of any SFC embedding method, leading to inefficient SFC placements, or even inhibit SFC embedding.**

To rectify this problem, we stress on the need for SFC graph transformation (SFC-GT), *i.e.,* explore the potential of SFC graph expansion prior to its embedding. SFC-GT aims at decomposing virtualized network functions (VNFs) into multiple instances with lower resource demands, facilitating their placement onto the NFVI. In this respect, we discuss the trade-off between *embedding flexibility* and *complexity*, in the context of SFC-GT. We formulate SFC-GT as a multi-objective optimization problem and design a mixed-integer linear program (MILP) to tackle it. Our simulation results demonstrate notable resource efficiency gains when SFC-GT is utilized prior to SFC embedding.

I. Introduction

Network Function Virtualization (NFV) decouples flow processing from specialized network devices, known as middleboxes, enabling the deployment of virtualized network functions (VNFs) on commodity servers with unprecedented flexibility and low cost. The deployment, configuration, and run-time management of VNFs entails various challenges (*e.g.*, resource allocation [1]–[5], service chaining [6], [7], and VNF scaling [8], [9]), which have spurred interest in the design and implementation of NFV orchestration platforms [3], [10].

A challenging problem pertaining to NFV orchestration is Service Function Chain (SFC) embedding. Existing methods optimize SFC embeddings with various objectives (*e.g.,* footprint minimization, inter-rack traffic minimization, load balancing), while adhering to capacity, flow preservation, and other constraints [1]–[5]. Irrespective of the optimization objective employed, the range of feasible SFC embedding solutions may be highly dependent on the condition of the underlying infrastructure, such as the utilization level and the resource fragmentation. For instance, a datacenter (DC) with highly utilized servers may not be in position to accommodate VNFs with high resource demands. In addition, a DC with significant degree of resource fragmentation may fail to meet a VNF co-location objective, leading to embeddings with large footprint, thereby, generating significant inter-rack traffic, which is undesirable for oversubscribed DC topologies.

ISBN 978-3-903176-39-3© 2021 IFIP

To circumvent the difficulty of generating efficient embeddings from a limited range of feasible solutions, we stress on the need for SFC graph transformation (SFC-GT), which can complement existing embedding methods, improving their resource efficiency, especially at conditions of the physical infrastructure that do not favor VNF placements (*e.g.,* high utilization, fragmentation). SFC-GT aims at the transformation of an initial SFC graph (*e.g.,* as supplied by a client), while maintaining the properties (*e.g.,* VNF order, total resource demands) of the original SFC graph. Opting for increased embedding flexibility, SFC-GT seeks to generate an expanded version of the original graph, *i.e.,* with additional instances for (some of) the VNFs. The decomposition of VNFs into multiple instances with lower resource demands: (i) can significantly increase the solution space, allowing for better embedding decisions, (ii) can reduce SFC embedding rejections, since smaller VNF instances fit more easily into a highly utilized or fragmented infrastructure, and (iii) can enable the placement of VNFs, whose resource demand exceeds the total capacity of a server (which is infeasible without SFC-GT).

The computation of extended SFC graphs is by no means a trivial task. First, the generation of multiple instances requires the careful positioning of additional nodes in the graph for traffic distribution among the instances, as well as additional edges. These entail significant modeling challenges, which we address in the problem formulation in Section V. Second and most importantly, the generation of an arbitrary number of instances could lead to significant resource overheads (*e.g.,* resource consumption of additional nodes, switch Ternary Content Addressable Memory (TCAM) consumption due to additional forwarding entries, server memory consumption due to VNF state replication), which could outweigh the gains in terms of embedding flexibility. Thereby, the most significant challenge of the envisaged SFC-GT is the tussle between embedding flexibility and complexity (which encompasses the resource overhead, amongst other aspects).

In this paper, we study the problem of optimized SFC transformation, investigating the most critical aspects of embedding flexibility and complexity, as well as their impact on SFC-GT. Our main contribution lies in the establishment of a holistic solution for the optimized transformation of SFC graphs. In this respect, we design a mixed-integer linear program (MILP) that computes SFC transformations, while balancing resource overheads with the extra flexibility afforded by additional

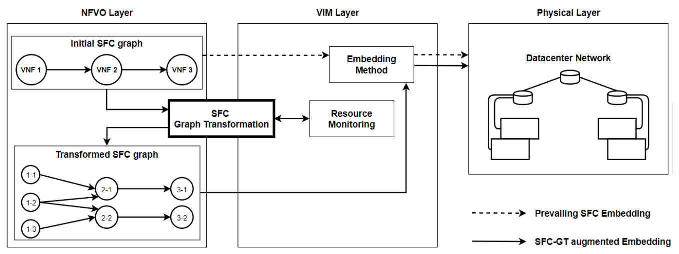


Fig. 1: *SFC-GT augmented* embedding approach.

VNF instances. Our SFC-GT MILP is complemented with a SFC embedding heuristic in order to assess the gains of SFC transformations in embedding efficiency. Our simulation results show notable resource efficiency gains, when SFC-GT is utilized, and also provide useful insights into the outcome of SFC transformation and the evolution of resource allocation across the infrastructure. To the best of our knowledge, a problem similar to SFC-GT has not been thoroughly studied in the literature.

The remainder of the paper is organized as follows. Section II outlines the proposed SFC-GT approach. In Section III, we define the required SFC models, whereas Section IV elaborates on the aspects of *embedding flexibility* and *complexity*, in the context of SFC-GT. Section V presents our SFC-GT MILP formulation, with detailed discussion of the objective function and constraints. In Section VI, we discuss our evaluation results, and Section VII highlights our conclusions.

II. SFC Graph Transformation Approach

SFC-GT alleviates the inefficiency of SFC embedding methods at conditions that restrict the embedding solution space, such as high utilization and/or high degree of resource fragmentation across the infrastructure. Under such conditions, a SFC embedder is restricted to a small set of feasible solutions, one of which has to be picked, although it may not comply with the embedding objective. Regardless of the degree of sophistication exhibited by the embedding method, there is very little that the embedder can do in such a case.

SFC-GT constitutes a viable approach to this problem. In particular, SFC-GT aspires to expand SFC graphs by decomposing VNFs into multiple instances. Each instance in the extended graph has a lower resource demand than the corresponding VNF in the initial graph. The resource demands of all instances of a specific VNF should sum up to the demand of the corresponding VNF (as expressed in the initial SFC graph). In our SFC-GT problem formulation (Section V), we discuss in detail all requirements for the extended graph. The main intuition behind the VNF instance decomposition is that VNF instances with lower resource demands can be more easily accommodated into a highly utilized or highly fragmented virtualized infrastructure (*e.g.,* datacenter). Fig. 1 illustrates a simple example of a SFC-GT. In particular, an initial SFC graph comprising three VNFs is transformed into an extended SFC graph, at which *VNF 1* has been decomposed into three instances, whereas *VNF 2* and *VNF 3* encompass two instances, each. The extended graph also contains additional edges that connect the instances of subsequent VNFs. Note that Fig. 1 merely provides a simplified view of the extended graph (*i.e.,* we have omitted additional nodes inserted before decomposed VNFs for traffic distribution among the VNF instances). We discuss in full detail all attributes of the SFC extended graph in Section III.

Since *embedding flexibility* increases with the number of VNF instances, one may expect that the SFC-GT should favor extended graphs with a multitude of instances. However, this is not the case, since additional VNF instances generate significant resource overheads that can outweigh the gains from the increased embedding flexibility. This resource overhead stems from various resource consumption aspects, such as the resources allocated for additional VNFs for traffic distribution among the VNF instances, switch TCAM consumption by the forwarding entries required by the additional edges in the extended graph, and the server memory consumption due to VNF state replication (we assume that VNF state is replicated among all running instances, as we explain in Section IV-C). We use the term *complexity* to accumulate these resource overheads, which we model in Section IV-C. Apparently, *embedding flexibility* and *complexity* are contradicting factors that both require optimization by SFC-GT. As such, the SFC-GT problem is far from trivial and requires careful attention in the modelling and balancing of these factors.

In Fig. 1, we depict an example of the application of SFC-GT in the context of NFV management and orchestration (MANO) frameworks. An initial SFC request is supplied to the NFV orchestrator (NFVO), which is, in turn, conveyed to a SFC-GT module that computes an optimal SFC transformation in the form of an extended SFC graph. The latter is sent to the Virtualized Infrastructure Manager (VIM) for its embedding and deployment on the virtualized infrastructure. Apparently, as shown in Fig. 1, SFC-GT is executed prior to the SFC embedding.

III. Extended Graph Model

Service Function Chain Request (SFC-R) model. We use a directed graph $G=(V,E)$ to express a SFC-R. The set of nodes V includes all VNFs i that are associated with a demand value on CPU, denoted by d_i. Edges between nodes i and j ($i,j \in V$) are expressed as $(i,j) \in E$, whereas the incoming traffic to VNF j is expressed with d^j.

Extended Service Function Chain (E-SFC) model. The E-SFC directed graph, denoted by $G^e=(V^e,E^e)$, is associated with a graph G that describes an arbitrary SFC-R, as discussed above, and is used to model the most expanded version of G. To this end, G^e generates multiple instances for each VNF, based on a pre-determined maximum number of instances for each particular VNF. A load balancer (LB) is inserted before the instances of a particular VNF for traffic splitting among the respective instances.

In more detail, given a SFC-R G comprising k VNFs ($k \in \mathbb{Z}, k \geq 2$), we define $m=\{m_1,...,m_k\}$, where m_i denotes the maximum number of instances permitted for the i^{th} VNF (*e.g.*, software licenses). Furthermore, we denote with v_i the

LB placed before VNF i, $i \in K = \{1, ..., k\}$, and $v_{i,j}$ the j^{th} instance of VNF i, $j \in K_i = \{1, ..., m_i\}$, $i \in K$. For convenience, we further use $B_i = \{v_i\}$ to express the unit set of the LB positioned before VNF i, and $V_i = \{v_{i,1}, ..., v_{i,m_i}\}$ to express the set of all potential VNF i instances. Thereby, the E-SFC graph G^e is described by the set of nodes $V^e = \bigcup_{i \in K} B_i \cup V_i$.

Subsequently, we distinguish between four types of edges between nodes in V^e. The first one refers to the set of edges between LBs and their successor VNF instances, and is expressed as $E_{BV} = \bigcup_{i \in K} \bigcup_{j \in K_i} (v_i, v_{i,j})$. Furthermore, we consider the edges between VNF instances and the LB placed behind the next VNF, denoted by $E_{VB} = \bigcup_{i \in K - \{k\}} \bigcup_{j \in K_i} (v_{i,j}, v_{i+1})$. In addition, we take into account the case at which two adjacent VNFs share the same number of instances; in this case, the insertion of a LB between the VNFs is omitted. To this end, we define $E_{VV} = \bigcup_{i \in K - \{k\}} \bigcup_{j \in min(K_i, K_{i+1})} (v_{i,j}, v_{i+1,j})$ to be the set of edges that explicitly connect the respective instances of adjacent VNFs, where:

$$min(K_i, K_{i+1}) = \begin{cases} K_i & |K_i| \leq |K_{i+1}| \\ K_{i+1} & \text{otherwise} \end{cases}$$

Last, if a VNF i ($i \in K - \{1\}$) comprises a single instance, there is no need to insert a LB before the respective VNF. As such, the instances of the $i-1^{th}$ VNF can be directly connected to this VNF. Such edges are expressed with the set $E_{V1} = \bigcup_{i \in K - \{k\}} \bigcup_{j \in K_i} (v_{i,j}, v_{i+1,1})$. Consequently, the set of edges that describe the extended graph G^e is the union of the four aforementioned sets of edges, *i.e.*, $E^e = E_{BV} \cup E_{VB} \cup E_{VV} \cup E_{V1}$.

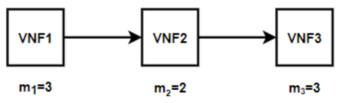
Fig. 2 illustrates an example of our E-SFC model. The initial SFC graph (Fig. 2a) consists of three VNFs. The E-SFC graph G^e, generated from G, is depicted in Fig. 2b. Each VNF is associated with an m-value that indicates the maximum number of instances that each respective VNF can encompass. For example, *VNF 1* can span at most three instances. LBs are interposed between consecutive VNFs in the E-SFC graphs, since for all consecutive pairs of VNFs the number of instances might be different. Essentially, the E-SFC graph defines the search space for SFC graph alternatives, which we term as feasible transformations (*e.g.,* Fig. 2c).

IV. SFC Transformation Problem

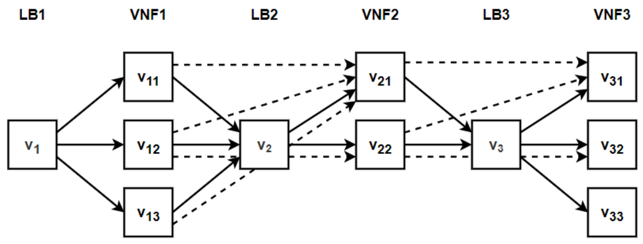
A. The SFC-GT Problem

Each SFC graph G adheres to a specific policy π (which describes the processing steps each packet traversing G is subject to) and is also associated with a desired processing capacity p (which is commonly expressed via VNF computing and bandwidth demands) in order to meet certain throughput rates or other service-oriented performance indicators (*e.g.,* service response time).

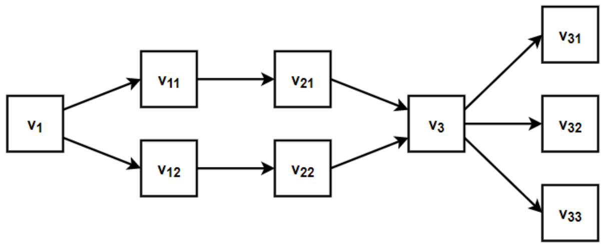
Definition 1. (Equivalent SFC graphs). Two SFC graphs G_1 and G_2 are equivalent if they adhere to the exact same policy, *i.e.*, $\pi_1 = \pi_2$, and they have an ε-similar processing capacity,



(a) Initial SFC graph G.



(b) Extended SFC graph G^e.



(c) A feasible transformation of G, G'.

Fig. 2: Different SFC graphs considered within the SFC-GT scope.

i.e., $p_1 = p_2 + \varepsilon, \varepsilon \approx 0$. We write $G_1 \sim G_2$ to denote that G_1 is equivalent to G_2.

Definition 2. (Feasible transformation). Given an initial SFC graph G and its associated extended graph G^e, a feasible transformation of G is every graph G', such that $G' \subset G^e$ and $G' \sim G$. *e.g.*, G' (Fig. 2c) is a feasible transformation of G (Fig. 2a).

Definition 3. (Set of feasible transformations). We define $FT(G)$ to be the set of all feasible transformations of G.

Definition 4. (Optimal transformation). Given an appropriate optimization function $f : (G, G', G_s) \mapsto \mathbb{R}$ (subject to minimization), where G is the initial graph, $G' \in FT(G)$, and G_s the substrate network graph, the optimal transformation of G is another graph G^*, such that $G^* \in FT(G)$, and $f(G, G^*, G_s) \leq f(G, G', G_s), \forall G' \in FT(G)$.

B. SFC Embedding Flexibility

As discussed in Section II, more expanded versions of SFC-Rs yield higher flexibility in terms of embedding, since there are more options for embedding smaller VNF instances onto a substrate network. As such, a SFC-GT method should opt for the transformation of the initial SFC graph in a way that greatly facilitates its embedding.

However, such a transformation is by no means straightforward. More precisely, the transformation needs to take into account not only the properties of the SFC graph, but also the condition (*i.e.,* utilization, fragmentation level) of the physical infrastructure. Therefore, we introduce the function $F : (G', G, G_s) \mapsto \mathbb{R}$, which takes as input a feasible transformation of G (*i.e.*, G'), the initial graph G, and the substrate network G_s, and is defined as follows:

$$F(G', G, G_s) = \phi(G_s) \cdot \frac{|V'| - |V|}{\sum_{i \in K} m_i - |V|} \tag{1}$$

where $\phi(G_s) \in [0, 1]$ denotes the percentage of servers in G_s with scarce resources, while $|V'|$ encompasses the total number of nodes (excluding LBs) that form G'. The fraction in Eq. (1) applies a min-max normalization on the value of $|V'|$.

For the trivial case of an entirely unutilized datacenter, the *embedding flexibility* of each $G' \in FT(G)$ will be the same, since $\phi(G_s)$ will be zero. For non-trivial cases (*i.e.,* with a certain amount of CPU utilization), $\phi(G_s) \neq 0$. As $\phi(G_s)$ increases, more weight will be given on the *embedding flexibility* of G', which is partially captured by $|V'|$. Note that $\phi(G_s)$ can be also adjusted to represent fragmentation [11], the average gap size [12], or other resource utilization indicators.

C. SFC Transformation Complexity

In principle, we consider several complexity aspects that can be attributed to a particular SFC graph structure. For instance, opting for G over G' (Figs. 2a and 2c) is expected to simplify service deployment and run-time service management, because of the smaller number of VNF instances that need to be spawned and managed. Furthermore, the number of instances affects the amount of state that a NFV orchestrator needs to maintain. In this work, we focus on what we deem as the most critical complexity aspect of SFC-GT, namely the resource overhead. In the following, we discuss the most significant factors that generate resource overhead. In this respect, assume that G represents the initial SFC graph, whereas $G' \in FT(G)$.

Load balancing. According to the E-SFC model, we insert virtualized LBs before VNFs to split the incoming traffic among the instances of each VNF. To prevent packet reordering, we rely on flow-based load balancing, which can be realized in a stateless manner (*e.g.,* similar to ECMP). With such a LB scheme, a flow aggregate can be easily split among a set of VNF instances with the same processing capacity. One possible implication of flow-based LB is that traffic may not be distributed evenly among the VNF instances, in the case of significant diversity in the flow sizes. This problem can be rectified by detecting large flows and directing them to separate instances [13]. Such LB mechanisms require additional functionality and state, which, in turn, would generate extra resource overhead.

Since LB per se is not the focus of this paper, we incorporate a stateless flow-based LB mechanism in our model. Since the main computationally-intensive task within such a LB is packet I/O, we have generated a resource profile driven from packet I/O computational requirements (*i.e.,* 1300 CPU cycles/packet, according to [14], [15]). The resource overhead introduced by the insertion of LBs in the E-SFC is computed as follows:

$$B(G', G) = \frac{|R'| - |R|}{|R_{max}| - |R|} \tag{2}$$

In Eq. (2), $|R'|$ and $|R|$ express the total CPU demands of G' and G, respectively, whereas $|R_{max}|$ denotes the CPU requirements of a feasible transformation of G that utilizes each and every LB from G^e (*i.e.*, such a graph would result in the maximum LB overhead).

TCAM consumption. VNF chaining requires the installation of flow entries in switch TCAMs [6], [7], [16]. A larger number of VNF instances is expected to lead to increased TCAM consumption. VNF instance co-location could reduce TCAM consumption; however, this is subject to capacity constraints which become more severe, as the utilization of the underlying infrastructure increases. We note that in commodity switches TCAM is usually limited (*i.e.,* few thousands of flow entries). The depletion of TCAM space can be potentially mitigated by the deployment of datapaths on commodity servers [17], [18]. However, this would lead to consumption of server capacity, which could have been utilized for VNF deployment instead.

In the context of SFC-GT, feasible transformations of a given graph G will typically encompass at least as many edges as G. With the largest embedding footprint (typical worst-case embedding scenario), each pair of adjacent nodes will be placed in servers within different racks and, as a consequence, each edge will be mapped onto a path that connects these two servers. In a two-layer datacenter fat-tree topology, such a path will span three switches (two top-of-the-rack and one core), hence, three flow entries will be required for each edge of the graph G. Observing the extended graph (Fig. 2b) derived from a simplistic initial graph, we expect a significant overhead in terms of TCAM consumption, in the case of a highly expanded graph. We denote this overhead as $M(G', G)$ and compute it:

$$M(G', G) = \frac{|E'| - |E|}{|E_{max}| - |E|} \tag{3}$$

where $|E'|$ and $|E|$ express the number of edges of G' and G, respectively, while $|E_{max}|$ corresponds to the maximum number of edges that can exist in a feasible graph of G.

VNF state. The majority of VNFs (*e.g.,* firewall, IDS, NAT) require internal state for their packet processing operations [8], [19]. Spawning additional instances of a stateful VNF requires, at the simplest case, the replication of their states across all instances, which leads to additional server memory consumption. Main memory consumption can be alleviated by correlating flows with running instances, such that each instance can maintain only a subset of the total state [9]. However, this yields higher complexity, since it requires the joint optimization of VNF instance placement and flow distribution. In our work, we account for the overhead stemming from state replication. The respective resource overhead is estimated as follows:

$$S(G', G) = \frac{s(G') - s(G)}{s(\underset{G_i \in FT(G)}{argmax} |V^i_{stateful}|) - s(G)} \tag{4}$$

where $s(G')$ and $s(G)$ express the number of stateful VNF instances that belong in G' and G, respectively. Additionally, we utilize $|V^i_{stateful}|$ to denote the number of stateful instances

TABLE I: Notations in the SFC-R, the E-SFC, and the MILP.

Symbol	Description
SFC-R	
V	the set of virtual nodes comprising a SFC
E	the set of virtual edges between virtual nodes
d_i	CPU demand of virtual node i
d^i	inbound traffic to VNF i
E-SFC	
m_i	maximum number of instances allowed for VNF i
K	index set of VNFs, *i.e.*, of V
K_i	index set of VNF i instances
v_i	the i^{th} LB node
$v_{i,j}$	the j^{th} instance of VNF i
B_i	the singleton comprising v_i
V_i	the set of $v_{i,j}$ instances
V^e	the set of virtual nodes comprising the extended graph
E_{BV}	the set of edges associated with v_i and $v_{i,j}$ nodes
E_{VB}	the set of edges associated with $v_{i,j}$ and v_{i+1} nodes
E_{VV}	the set of edges associated with $v_{i,j}$ and $v_{i+1,j}$ nodes
E_{V1}	the set of edges associated with $v_{i,j}$ and $v_{i+1,1}$ nodes
E^e	the set of virtual edges comprising the extended graph
MILP	
x_i	assignment of node v_i
$x_{i,j}$	assignment of node $v_{i,j}$
$r_{i,j}$	CPU demand assigned to the j^{th} instance of VNF i
$z^i_{i,j}$	assignment of edge $(v_i, v_{i,j}) \in E_{BV}$
$r^i_{i,j}$	resource demand assigned to edge $(v_i, v_{i,j})$
$z^{i,j}_{i+1}$	assignment of edge $(v_{i,j}, v_{i+1}) \in E_{VB}$
$r^{i,j}_{i+1}$	resource demand assigned to edge $(v_{i,j}, v_{i+1})$
$z^{i,j}_{i+1,j}$	assignment of edge $(v_{i,j}, v_{i+1,j}) \in E_{VV}$
$r^{i,j}_{i+1,j}$	resource demand assigned to edge $(v_{i,j}, v_{i+1,j})$
$z^{i,j}_{i+1,1}$	assignment of edge $(v_{i,j}, v_{i+1,1}) \in E_{V1}$
$r^{i,j}_{i+1,1}$	resource demand assigned to edge $(v_{i,j}, v_{i+1,1})$

in graph $G_i \in FT(G)$. Therefore, $s(\underset{G_i \in FT(G)}{argmax} |V^i_{stateful}|)$ is the maximum number of stateful instances that can exist in a feasible transformation of G.

Essentially, Eq. (2), (3) and (4) apply a min-max normalization on the first terms of their nominators; thus, they share the same co-domain, which is $[0,1]$. Since the resource overhead is considered to stem from load balancing, TCAM consumption, and VNF state replication, it is computed as:

$$C(G',G) = \alpha \cdot B(G',G) + \beta \cdot M(G',G) + \gamma \cdot S(G',G) \qquad (5)$$

where $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$, $\alpha + \beta + \gamma = 1$, and $C: \ (G',G) \mapsto [0,1]$.

V. Optimized SFC Transformation Formulation

A. Core Variables

We initially introduce the core variables of our MILP program, which are practically associated with the assignment of nodes and edges from G^e, as well as the assignment of the respective resources. We denote by x the binary variables that indicate the assignment of nodes, and by z the binary variables that indicate the assignment of edges between nodes. In particular, x_i refers to the assignment of v_i, *i.e.*, the i^{th} LB, whereas $x_{i,j}$ denotes the assignment of $v_{i,j}$, *i.e.*, the j^{th} instance of VNF i. Similarly, $z^i_{i,j}$ indicates the assignment of the edge that connects v_i and $v_{i,j}$, $z^{i,j}_{i+1}$ expresses the assignment of the edge that connects $v_{i,j}$ and v_{i+1}, and, last, $z^{i,j}_{i+1,j}$ and $z^{i,j}_{i+1,1}$ refer to edges in E_{VV} and E_{V1}, respectively. Each of the aforementioned variables (except from the variables x_i) is associated with an r-variable that expresses the amount of resources assigned to the corresponding virtual element. For example, $r_{0,2}$ refers to the CPU resources assigned to $v_{0,2}$, whereas $r^{0,2}_1$ represents the bandwidth requirements assigned to the link $(v_{0,2}, v_1)$.

B. Node and Edge Assignment Constraints

Reduction of the solution space. We assume that instances of a particular VNF have the same processing capacity. Thus, according to the E-SFC model, there are many solutions that essentially express the same graph. For instance, the graph $v_{1,1} \rightarrow v_{2,1} \rightarrow v_{3,1}$ is the same as $v_{1,3} \rightarrow v_{2,1} \rightarrow v_{3,1}$, *i.e.,* both graphs encompass the same number of instances for each VNF (see Fig. 2b). Constraint (6) is defined to ensure that VNF instances are selected in ascending order. This reduces the solution space and alleviates the solver from evaluating solutions with the same efficiency.

$$x_{i,j+1} \leq x_{i,j} \quad \forall j \in K_i - \{m_i\}, \forall i \in K \qquad (6)$$

At least one instance of each VNF is selected. Considering Eq. (6), this constraint is defined as shown in Eq. (7).

$$\sum_{i \in K} x_{i,1} = k \qquad (7)$$

LBs are not placed before VNFs with a singe-instance. This constraint, which eliminates unnecessary nodes, is defined as:

$$x_i + \varepsilon \leq \sum_{j \in K_i} x_{i,j} \quad \forall i \in K, \varepsilon \in (0,1) \qquad (8)$$

Assigned nodes enforce the selection of links, and vice-versa (Case: LB$_i$ $\rightarrow$ VNF$_i$). Constraint (9) ensures that a virtual link is established between v_i and the selected v_{ij}:

$$x_i \cdot x_{i,j} = z^i_{i,j} \quad \forall (v_i, v_{i,j}) \in E_{BV} \qquad (9)$$

Since Eq. (9) is a non-linear constraint and all of the associated variables are binary, Eq. (9) can be linearized with the following constraints:

$$z^i_{i,j} \leq x_i \quad \forall (v_i, v_{i,j}) \in E_{BV} \qquad (10)$$
$$z^i_{i,j} \leq x_{i,j} \quad \forall (v_i, v_{i,j}) \in E_{BV} \qquad (11)$$
$$x_i + x_{i,j} - 1 \leq z^i_{i,j} \quad \forall (v_i, v_{i,j}) \in E_{BV} \qquad (12)$$

Assigned nodes enforce the selection of links, and vice-versa (Case: VNF$_i$ $\rightarrow$ LB$_{i+1}$). In case a LB is placed before VNF $i+1$ (*i.e.*, $x_{i+1} = 1$), the selected instances $v_{i,j}$ of VNF i should be connected to it. This is defined as:

$$x_{i+1} \cdot x_{i,j} = z^{i,j}_{i+1} \quad \forall (v_{i,j}, v_{i+1}) \in E_{VB} \qquad (13)$$

We employ the previous logic, *i.e.*, similar to constraints (10) - (12), in order to linearize Eq. (13).

LBs are not placed between adjacent VNFs with the same number of instances. This constraint enables the creation of parallel paths within a SFC graph, thus, allowing for more flexible (in terms of embedding) and robust graph structures. This is formulated via the following expression:

$$\sum_{j\in K_i} x_{i,j} = \sum_{j\in K_{i+1}} x_{i+1,j} \Rightarrow x_{i+1} = 0 \quad \forall i \in K - \{k\} \tag{14}$$

which certainly does not conform to LP modelling requirements. To this end, we introduce the integer variables δ^{i}_{i+1}, δ^{i+}_{i+1}, δ^{i-}_{i+1}, and the binary variables y^{i}_{i+1}, as follows:

$$\delta^{i}_{i+1} = \sum_{j\in K_i} x_{i,j} - \sum_{j\in K_{i+1}} x_{i+1,j} \quad \forall i \in K - \{k\} \tag{15}$$

$$\delta^{i}_{i+1} = \delta^{i+}_{i+1} - \delta^{i-}_{i+1} \quad \forall i \in K - \{k\} \tag{16}$$

$$0 \le \delta^{i+}_{i+1} \le max(m_i, m_{i+1}) \cdot y^{i}_{i+1} \quad \forall i \in K - \{k\} \tag{17}$$

$$0 \le \delta^{i-}_{i+1} \le max(m_i, m_{i+1}) \cdot (1 - y^{i}_{i+1}) \quad \forall i \in K - \{k\} \tag{18}$$

$$x_{i+1} \le \delta^{i+}_{i+1} + \delta^{i-}_{i+1} \quad \forall i \in K - \{k\} \tag{19}$$

Essentially, constraints (15) to (18) impose that $\delta^{+} + \delta^{-} = |\delta| = |\sum_{j\in K_i} x_{i,j} - \sum_{j\in K_{i+1}} x_{i+1,j}|$. Consequently, Eq. (19) implies Eq. (14).

Assigned nodes enforce the selection of links, and vice-versa (Case: VNF$_i$ $\rightarrow$ VNF$_{i+1}$). It is further required to enforce the selection of edges that directly connect the respective instances of two adjacent VNFs, in case these are split into the same number of instances. Recall that the latter is implied by $\delta^{i+}_{i+1} + \delta^{i-}_{i+1} = 0$ (see constraint (14)). To this end, we introduce the following expression in our formulation:

$$\begin{aligned}(x_{i,j} = 1 \wedge x_{i+1,j} = 1) \wedge \delta^{i+}_{i+1} + \delta^{i-}_{i+1} = 0 \\ \iff z^{i,j}_{i+1,j} = 1 \quad \forall j \in K_i, \forall i \in K - \{k\}\end{aligned} \tag{20}$$

which can be easily linearized by the insertion of $t^{i,j}_{i+1,j}$ binary variables to hold the product of $x_{i,j}$ and $x_{i+1,j}$ (in a similar manner to the linearization of Eq. (9) and Eq (13)). In particular, the following expressions linearize constraint (20):

$$\begin{aligned}t^{i,j}_{i+1,j} - \delta^{i+}_{i+1} - \delta^{i-}_{i+1} \le max(m_i, m_{i+1}) \cdot z^{i,j}_{i+1,j} \\ \forall (v_{i,j}, v_{i+1,j}) \in E_{VV}\end{aligned} \tag{21}$$

$$\begin{aligned}\delta^{i+}_{i+1} + \delta^{i-}_{i+1} - t^{i,j}_{i+1,j} + \varepsilon \le max(m_i, m_{i+1}) \cdot (1 - z^{i,j}_{i+1,j}) \\ \forall (v_{i,j}, v_{i+1,j}) \in E_{VV}, \varepsilon \in (0,1)\end{aligned} \tag{22}$$

Assigned nodes enforce the selection of links, and vice-versa (Case: VNF$_i$ $\rightarrow$ VNF$_{i+1,1}$). The following constraint enforces the required binding between the x and z variables, when there is only a single instance of VNF $i+1$, *i.e.*, $x_{i+1,2} = 0$ (see constraint (6)). In this case, according to constraint (8), we do not place a LB before the single instance. As such, the assigned $v_{i,j}$ instances, *i.e.*, those for which $x_{i,j} = 1$, should be directly connected to the $v_{i+1,1}$ node. This is expressed as follows:

$$\begin{aligned}x_{i+1,2} = 0 \wedge x_{i,j} = 1 \iff z^{i,j}_{i+1,1} = 1 \\ \forall j \in K_i, \forall i \in K - \{k\}\end{aligned} \tag{23}$$

which is equivalent to the following linear expressions:

$$x_{i,j} - x_{i+1,2} \le z^{i,j}_{i+1,1} \quad \forall (v_{i,j}, v_{i+1,1}) \in E_{V1} \tag{24}$$

$$z^{i,j}_{i+1,1} \le x_{i,j} \quad \forall (v_{i,j}, v_{i+1,1}) \in E_{V1} \tag{25}$$

$$z^{i,j}_{i+1,1} \le 1 - x_{i+1,2} \quad \forall (v_{i,j}, v_{i+1,1}) \in E_{V1} \tag{26}$$

Place a LB before the first VNF with multiple instances. The first VNF in a SFC is treated as a special case, since the placement of a LB is required before the VNF, in case it encompasses multiple instances, *i.e.*, $\sum_{j\in K_1} x_{1,j} \ge 2$. In LP terms, this is formulated as:

$$\sum_{j\in K_1} x_{1,j} - 2 + \varepsilon \le m_1 \cdot x_1 \quad \varepsilon \in (0,1) \tag{27}$$

Place a LB between two VNFs with different number of instances and when the second VNF has multiple instances. According to the previous discussions, we formulate this constraint as:

$$\begin{aligned}\delta^{i+}_{i+1} + \delta^{i-}_{i+1} \ge 1 \wedge x_{i+1,2} = 1 \Rightarrow x_{i+1} = 1 \\ \forall i \in K - \{k\}\end{aligned} \tag{28}$$

which is equivalent to the linear constraints:

$$x_{i+1} \le x_{i+1,2} \cdot max(m_i, m_{i+1}) \quad \forall i \in K - \{k\} \tag{29}$$

$$\begin{aligned}\delta^{i+}_{i+1} + \delta^{i-}_{i+1} - (1 - x_{i+1,2}) \cdot max(m_i, m_{i+1}) \\ \le x_{i+1} \cdot max(m_i, m_{i+1}) \quad \forall i \in K - \{k\}\end{aligned} \tag{30}$$

along with constraint (19).

C. Resource Assignment Constraints

Resource demands for VNF instances. Apparently, the resource demands of all VNF instances must sum up to the initial resource demands of the respective VNF. Recall that we opt for VNF instances with the same processing capacity, hence, their resource demands will be equal.

$$\sum_{j\in K_i} r_{i,j} \cdot x_{i,j} = d_i \quad \forall i \in K \tag{31}$$

$$\begin{aligned}x_{i,j+1} = 1 \Rightarrow r_{i,j} = r_{i,j+1} \\ \forall j \in K_i - \{m_i\}, \forall i \in K\end{aligned} \tag{32}$$

Yet, we should pay attention to the product of r and x variables in order to avoid non-linearity. To this end, we introduce $h_{i,j}$, which is a positive continuous variable adhering to:

$$h_{i,j} \le x_{i,j} \cdot d_i \quad \forall i \in K \tag{33}$$

$$r_{i,j} - (1 - x_{i,j}) \cdot d_i \le h_{i,j} \quad \forall i \in K \tag{34}$$

$$h_{i,j} \le r_{i,j} \quad \forall i \in K \tag{35}$$

Constraints (33) - (35) ensure that $h_{i,j}$ holds the product $r_{i,j} \cdot x_{i,j}$. Therefore, Eq. (31) can be now expressed as:

$$\sum_{j\in K_i} h_{i,j} = d_i \quad \forall i \in K \tag{36}$$

while Eq. (32) can be enforced by the following expressions in conjunction with constraint (6):

$$r_{i,j+1} \leq r_{i,j} \quad \forall j \in K_i - \{m_i\}, \forall i \in K \tag{37}$$

$$r_{i,j} - r_{i,j+1} \leq d_i \cdot (1 - x_{i,j+1}) \quad \forall j \in K_i - \{m_i\}, \forall i \in K \tag{38}$$

$$r_{i,j+1} \leq d_i \cdot x_{i,j+1} \quad \forall j \in K_i - \{m_i\}, \forall i \in K \tag{39}$$

Resource demands for edges departing from VNFs. The bandwidth demands of edges that depart from VNF i instances must sum up to d^{i+1}, *i.e.*, the total ingress traffic to VNF $i+1$. To enforce this, we introduce the following constraints:

$$\sum_{j \in K_i} (z^{i,j}_{i+1} \cdot r^{i,j}_{i+1} + z^{i,j}_{i+1,1} \cdot r^{i,j}_{i+1,1}) + \sum_{j \in min(K_i, K_{i+1})} z^{i,j}_{i+1,j} \cdot r^{i,j}_{i+1,j} = d^{i+1} \quad \forall i \in K - \{k\} \tag{40}$$

The individual products of z and r variables can be linearized in the same manner with Eqs. (33) - (35) for Eq. (31). However, we omit these nine constraints (three per product), due to space limitations. Since we employ constraint (32), we distribute the required bandwidth evenly among all edges, *i.e.*, edges that depart from VNF i have equal resource demands. This is captured by the following expressions:

$$z^{i,j+1}_{i+1} = 1 \Rightarrow r^{i,j}_{i+1} = r^{i,j+1}_{i+1} \quad \forall j \in K_i - \{m_i\}, \forall i \in K - \{k\} \tag{41}$$

$$z^{i,j+1}_{i+1,1} = 1 \Rightarrow r^{i,j}_{i+1,1} = r^{i,j+1}_{i+1,1} \quad \forall j \in K_i - \{m_i\}, \forall i \in K - \{k\} \tag{42}$$

$$z^{i,j+1}_{i+1,j+1} = 1 \Rightarrow r^{i,j}_{i+1,j} = r^{i,j+1}_{i+1,j+1} \quad \forall j \in min(K_i, K_{i+1}) - \{min(m_i, m_{i+1})\}, \forall i \in K - \{k\} \tag{43}$$

Each one of the Eqs. (41), (42), and (43) can be applied in our MILP program through triplets of constraints (omitted for brevity), similar to Eqs. (37), (38), and (39).

Resource demands for edges departing from LBs. We further consider the bandwidth demands of edges that depart from the assigned LBs. In this respect, the following expression implies flow conservation, *i.e.,* the amount of inbound traffic to the LB is equal to the total amount of outbound traffic. This is formulated as:

$$\sum_{j \in K_i} z^i_{i,j} \cdot r^i_{i,j} = d^i \cdot z^i_{i,1}, \quad \forall i \in K \tag{44}$$

We linearize Eq. (44) by incorporating three additional constraints (similar to Eqs. (33) - (35)) in order to model the product between $z^i_{i,j}$ and $r^i_{i,j}$, whereas another triplet of constraints, in a similar manner to Eqs. (37) - (39), is utilized to enforce equal bandwidth demands among the edges that depart from LBs.

D. Objective Function

The objective of the SFC transformation problem is as follows. Given an SFC-R G, a substrate network G_s, and its resource utilization state $\phi(G_s)$, identify $G^* \in FT(G)$, such that $f(G^*, G, G_s) \leq f(G', G, G_s), \forall G' \in FT(G)$, where

$$f(G', G, G_s) = (1 - \phi(G_s)) \cdot C(G', G) - F(G', G, G_s) \tag{45}$$

Recall that, as explained in Section IV, C and F are contradictory; thus, our MILP seeks *Pareto-optimal* solutions (specifically, it tries to minimize the *complexity*, while maximizing the *embedding flexibility* of the transformation). Furthermore, since $F(G', G, G_s)$ depends heavily on $\phi(G_s)$, the value of Eq. (45) will be mainly driven by $C(G', G)$ whenever $\phi(G_s)$ is close to zero. However, as $\phi(G_s)$ deviates from zero, $F(G', G, G_s)$ will have a greater impact on the fitness score of candidate solutions.

VI. Evaluation

In this section, we aim at quantifying the gains from our proposed SFC transformation method. To this end, we utilize a SFC embedding method in the following two evaluation scenarios: (i) our SFC-GT scheme generates an optimized transformation of the initial SFC-R, which is subsequently conveyed to the SFC embedding method (termed as *GT-augmented* embedding), and (ii) the initial SFC-R is processed directly by the SFC embedding method, without any SFC transformation (termed as *baseline* embedding).

The SFC embedding method is essentially a heuristic that computes SFC embeddings using the following steps. The heuristic maps the VNFs of the SFC in a sequential manner, starting with the first VNF of the SFC. Initially, it ranks the racks of the datacenter in descending order, according to their available ToR to core link capacity, and also ranks the servers of the current rack in descending order, according to their residual CPU capacity. Subsequently, the heuristic places the first VNF in the top ranked server, and prioritizes the assignment of adjacent VNFs in the same server. If such assignments are infeasible, the heuristic seeks another server of the current rack; otherwise, it repeats the process at the ranked servers of the subsequent rack, until each VNF has been successfully placed. In case the placement of any VNF (or link) in the SFC fails, the embedding request is rejected.

A. Evaluation Environment

We have developed a simulation environment for SFC embedding evaluations in Python. The SFC-GT MILP is implemented with the Gurobi. All evaluations are carried out on a simulated two-layer fat-tree datacenter network topology. The datacenter comprises 200 servers, organized in ten racks. The corresponding top-of-the-rack (ToR) switches are interconnected through five core switches. The intra-rack and inter-rack links have capacity of 1 Gbps and 10 Gbps, respectively, and the CPU capacity of each server is set to 7.2 GHz.

Each SFC-R consists of three to eight VNFs, whereas the CPU demand of each VNF in the SFC-R lies between 10 to 50% (with 10% step) of 7.2 GHz. The inbound traffic in the

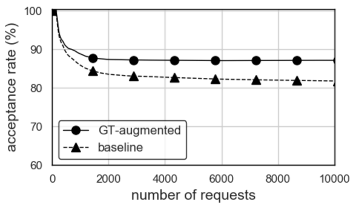


Fig. 3: Request acceptance rate.

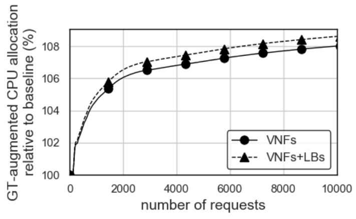


Fig. 4: CPU utilization of *GT-augmented*, in relation to *baseline*.

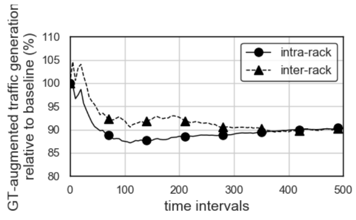


Fig. 5: Generated traffic by *GT-augmented*, in relation to *baseline*.

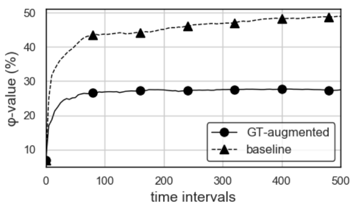


Fig. 6: ϕ-values of *GT-augmented* and *baseline*.

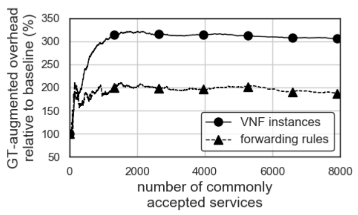


Fig. 7: VNF instances and forwarding rules of *GT-augmented*, in relation to *baseline*.

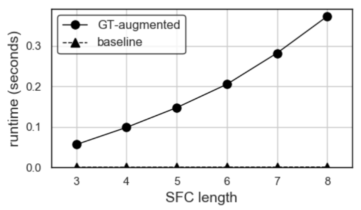


Fig. 8: Average solver runtime for diverse SFC lengths.

SFC lies anywhere between 50 and 200 Mbps. Concerning the maximum number of instances that each VNF can span (*i.e.*, m_i-values), we set these to five, while there is 80% probability that a VNF is stateful (otherwise, it is stateless). Furthermore, $\phi(G_s)$ indicates the percentage of servers with available CPU between 3% and 15%, whereas for the complexity metric, we set α, β, and γ to 0.4, 0.4, and 0.2, respectively. In future work, we will investigate more precise values for these parameters based on experimentation with VIMs, such as OpenStack.

In addition, we account for a set of n discrete time intervals $T = \{1,2,...,n\}$. At each time interval, N SFC-Rs arrive sequentially, where N follows a Poisson distribution with $\lambda = 20$. Each NS comes with a lifespan k. Therefore, if a NS is embedded at time $t \in T$, it will expire at time $t+k \in T$. During our simulations, k lies within [3,10] (randomly), whereas n is set to 500.

B. Evaluation Results

Initially, we compare the request acceptance rate achieved by the *GT-augmented* method and the *baseline*. This metric expresses the ratio of the number of successfully embedded SFC-Rs over the total number of SFC-Rs. According to Fig. 3, *GT-augmented* admits 87% of the requests, whereas the respective percentage for the *baseline* is 82%. Further examination of the acceptance rate results uncovers that all rejections of *GT-augmented* are attributed to lack of CPU resources, whereas a small fraction of the *baseline* rejections is caused by lack of bandwidth, as well.

As expected, the higher acceptance rate of the proposed method is translated into higher resource efficiency, as shown in Fig. 4. In this plot, we compute the moving average of the fractions CPU_i^1/CPU_i^2, where CPU_i^1 is the sum of the CPU assigned to SFCs, until request i by the *GT-augmented* method (the denominator captures the same value for the *baseline*). Recall that a graph transformation may introduce further CPU requirements, because of the addition of LBs. The reasonable assumption is that such resources are not monetized by the InPs, who may opt for utilizing them in favor of higher resource efficiency. That said, Fig. 4 indicates that the *GT-augmented* algorithm manages to assign 9% more CPU compared to the baseline, while the net gain (*i.e.*, resources for VNFs, which are indeed monetized) is 8%.

Fig. 5 illustrates the relative efficiency of the proposed method against the *baseline* in terms of generated traffic, subdivided into intra- and inter-rack. In this plot, the respective values are computed per time interval (which is per $\approx$20 SFC-Rs), thereby, covering the whole range of the simulation. While initially both curves tend to fluctuate, they eventually reach a steady state, which is the same for both DC topology levels, *i.e.,* at 90% of the traffic generated by the *baseline*. Apparently, this indicates higher efficiency for the proposed method, especially at the inter-rack level, where bandwidth conservation is crucial. At first, bandwidth conservation through VNF decomposition (into multiple instances) may not be expected, especially since more emphasis was given (narrative-wise) on relaxing the CPU requirements for enhanced *embedding flexibility*. Nevertheless, SFC-GT may indeed generate less traffic, since VNF decomposition opens up opportunities for co-locating more communicating SFC components into the same server or rack. For instance, each one of the links $v_{11} \rightarrow v_{21}$ and $v_{12} \rightarrow v_{22}$ in Fig. 2c yields half of the bandwidth requirements compared to the link $VNF1 \rightarrow VNF2$ in Fig. 2a. Therefore, if $VNF1$ and $VNF2$ are not placed on the same server, the whole required bandwidth will be allocated from the corresponding DC links. In contrast, assume that the *GT-augmented* method places only v_{12} and v_{22} on different servers. This would result in only half of the bandwidth consumption observed in the previous case (*i.e.,* without employing *GT-augmented*).

To gain further insights on the behavior of the two methods, we examine the condition of the underlying DC, quantified by the ϕ-values, *i.e.*, the percentage of servers with available CPU between 3% and 15%, which we use as a rough approximation for CPU fragmentation. According to Fig. 6, ϕ reaches 50% with the *baseline* (with a tendency to increase), whereas the respective value with the *GT-augmented* is significantly lower. Additional results (which are omitted due to space limitations) indicate that, at steady state, the *baseline* utilizes 80% of the servers' CPU (the respective value with *GT-augmented* is 90%). This, along with the measured high fragmentation, implies that a considerable portion of the residual CPU is scattered in resource blocks of size 3% to 15%, a fact that highly restricts the embedding options of SFCs, eventually reducing the efficiency of the *baseline*.

Recall that SFC-GT seeks a balance between maximizing the *embedding flexibility* and minimizing the *complexity* of graph transformations. So far, our evaluation results corroborate the gains of SFC-GT in terms of *embedding flexibility*. With respect to *complexity*, Fig. 7 illustrates the generated VNF instances, as well as the required forwarding rules for the proposed method, in relation to the initial service graph and its mapping by the *baseline*. For this comparison, the successful mapping of the respective graphs (*i.e.*, transformed and initial) is crucial, hence, we only account for the commonly accepted SFC-Rs by both methods. In this respect, we observe x3 increase in the number of VNF instances, and x2 increase in terms of forwarding rules. Practically, this means that the transformed graphs embedded by *GT-augmented* comprise of triple the nodes of the initial graph, while their chaining requires twice as much forwarding rules on the switches, on average. Given that m_i-values are set to 5, this result indicates an achieved balance in terms of VNF instances. In terms of forwarding rules, steering traffic across more VNF instances inevitably increases switch TCAM consumption. In future work, we will seek to alleviate this TCAM consumption overhead, by coupling SFC-GT with source routing in an experimental environment.

Finally, we compare the solver runtime between the *GT-augmented* and the *baseline*. According to Fig. 8, the runtime of *GT-augmented* exhibits an exponential growth, in relation to the problem size (*i.e.,* SFC length). However, for reasonable SFC lengths and values of m_i, the *GT-augmented* runtime is bounded below 0.4 sec, which indicates its feasibility in realistic embedding scenarios.

VII. Conclusions

In this paper, we tackled the challenging problem of SFC-GT in order to empower SFC embedders to perform more efficient VNF placements. To this end, we presented a SFC-GT problem formulation that captures various aspects of embedding flexibility and resource overheads, as well as the design of a MILP that seeks *Pareto-optimal* solutions for SFC transformations. Our evaluations indicate that our multi-objective optimization adapts SFC graphs according to the prevailing conditions, leading to enhanced resource efficiency, especially for medium and high utilization levels.

VIII. Acknowledgments

This work is supported by the MESON (Optimized Edge Slice Orchestration) project, co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T1EDK-02947).

References

[1] Dietrich *et al.*, "Network function placement on virtualized cellular cores," in *IEEE COMSNETS*, 2017.

[2] Tomassilli *et al.*, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *IEEE INFOCOM*, 2018.

[3] Palkar *et al.*, "E2: a framework for nfv applications," in *ACM SOSR*, 2015.

[4] Dietrich *et al.*, "Multi-provider service chain embedding with nestor," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 91–105, 2017.

[5] A. Pentelas *et al.*, "Network service embedding across multiple resource dimensions," *IEEE Transactions on Network and Service Management*, 2020.

[6] Qazi *et al.*, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4, 2013, pp. 27–38.

[7] S. K. Fayazbakhsh *et al.*, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *USENIX NSDI*, Apr. 2014.

[8] Gember-Jacobson *et al.*, "Opennf: Enabling innovation in network function control," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 163–174.

[9] Cao *et al.*, "Distributed data deluge (d3): efficient state management for virtualized network functions," in *IEEE INFOCOM SWFAN*, 2016.

[10] Kourtis *et al.*, "T-nova: An open-source mano stack for nfv infrastructures," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 586–602, 2017.

[11] J. Gehr and J. Schneider, "Measuring fragmentation of two-dimensional resources applied to advance reservation grid scheduling," in *IEEE/ACM CCGrid*, 2009.

[12] L. Tom *et al.*, "Improving grid resource usage: Metrics for measuring fragmentation," in *IEEE/ACM CCGrid*, 2012.

[13] M. Al-Fares *et al.*, "Hedera: Dynamic flow scheduling for data center networks," in *USENIX NSDI*, 2010.

[14] M. Dobrescu *et al.*, "Routebricks: exploiting parallelism to scale software routers," in *ACM SOSP*, 2009.

[15] A. Abujoda and P. Papadimitriou, "Profiling packet processing workloads on commodity servers," in *IFIP WWIC*, 2013.

[16] C. Papagianni *et al.*, "Towards reduced-state service chaining with source routing," in *IEEE CNSM*, 2018.

[17] N. Sarrar *et al.*, "Leveraging zipf's law for traffic offloading," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 1, p. 16–22, Jan. 2012.

[18] Z. Bozakov and P. Papadimitriou, "Openvroute: An open architecture for high-performance programmable virtual routers," in *IEEE HPSR*, 2013.

[19] S. Rajagopalan *et al.*, "Split/merge: System support for elastic execution in virtual middleboxes," in *USENIX NSDI*, Apr. 2013.

European Union
European Regional Development Fund

HELLENIC REPUBLIC
MINISTRY OF ECONOMY & DEVELOPMENT
SPECIAL SECRETARY FOR ERDF & CF
MANAGING AUTHORITY OF EPAnEK

EPAnEK 2014-2020
OPERATIONAL PROGRAMME
COMPETITIVENESS ENTREPRENEURSHIP INNOVATION

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη
Partnership Agreement 2014 - 2020

Co-financed by Greece and the European Union