A Novel Method for SFC Segmentation in Edge-Cloud Environments

Anselmo Luiz Éden Battisti
MídiaCom Lab, Institute of Computing
UFF - Niterói/RJ, Brazil
anselmo@midiacom.uff.br

Flavia C. Delicato
MídiaCom Lab, Institute of Computing
UFF - Niterói/RJ, Brazil
fdelicato@ic.uff.br

Débora Christina Muchaluat-Saade
MídiaCom Lab, Institute of Computing
UFF - Niterói/RJ, Brazil
debora@midiacom.uff.br

***Abstract*—A Service Function Chain (SFC) is a chain of Virtual Network Functions (VNFs). SFCs enable orchestration of complex network services in edge-cloud environments. To execute an SFC in a multi-domain environments, it is necessary to split the SFC into small parts named segments. This paper presents the Distributed Segmentation Strategy (DSS), a novel approach to segmenting SFCs. The SFC segmentation problem involves dividing an SFC into smaller parts to optimize the deployment and operation of network services across multiple domains. Our method allows for configurable segmentation strategies, enabling different SFC placement methods to tailor the segmentation process according to their specific requirements. Experimental results demonstrate that DSS enhances performance based on the SFC placement criteria, making it a valuable contribution to Network Function Virtualization (NFV) environments.**

***Index Terms*—Service Function Chain, SFC placement, SFC segmentation**

I. Introduction

Virtual Network Functions (VNFs) allow a software-based implementation of network functions that are traditionally executed in dedicated hardware [1]. These VNFs enable flexible deployment and scalability by running on commodity hardware, thus reducing costs and enhancing network management efficiency. Service Function Chains (SFCs) are sequences of VNFs, arranged to process network traffic in a predefined order to provide complex network services [2].

To execute an SFC, each VNF that makes up the chain must be instantiated on physical network nodes for execution. The process of selecting which physical nodes will execute each VNF is defined as the SFC placement problem [3]. This problem is complex and substantial research has already been done to address it [4].

The SFC placement problem consists of determining the optimal deployment of VNFs within an SFC to meet various performance, cost, and policy requirements. This problem can be approached in both centralized and decentralized fashions [4]. In a centralized approach, a single orchestrator is responsible for the global view and control of the entire network, making placement decisions based on a comprehensive understanding of the available resources and network topology. This approach can efficiently optimize resource allocation, but it may suffer from scalability issues and a single point of failure [5]. On the other hand, a decentralized approach distributes the decision-making process across multiple nodes or domains, each making local placement decisions based on partial knowledge of the network and computational resources. This feature is relevant for edge-cloud environments, as it allows nodes with varying capacities to collaborate in the execution of the corresponding SFC [6]. Although this method improves scalability and resilience by avoiding central bottlenecks and a single point of failure, it can lead to suboptimal resource utilization due to the lack of a global view. Methods for solving the SFC placement problem in multi-domain environments are composed of multiple steps, including resolving an SFC segmentation problem.

The SFC segmentation problem divides an SFC into smaller parts, called SFC segments, or only segments for simplicity [7], [8]. Segments are portions of an SFC that are confined within a single domain. These segments consist of VNFs that are interconnected and operate together to perform specific network functions within the limits of their segment. Each segment is designed to handle a subset of the overall SFC operations, ensuring efficient and localized processing of network traffic. This segmentation facilitates better resource allocation, reduces latency, and improves the overall efficiency and manageability of network functions by leveraging localized optimizations and interconnecting segments through secure tunneling technologies such as VPNs or VXLANs [9]. By enabling distributed processing and reducing inter-domain complexity, SFC segmentation plays a crucial role in the SFC placement process in multi-domain environments.

In this work, we present a novel method called the Distributed Segmentation Strategy (DSS). This method addresses the SFC segmentation problem in edge-cloud environments, using an approach inspired by the "stars and bars" problem, as detailed in [10]. Our approach significantly advances the field, offering the following advantages: i) it can be integrated seamlessly with many distributed SFC placement methods that need SFC segmentation in multi-domain environments, as our proposed heuristic is modular and decouples segmentation from the placement process, ii) combined with an SFC placement method, it can improve the efficiency of SFC deployment across multi-domain environments, iii) the time required to generate the SFC segments with our approach has a minimal impact on the overall SFC placement time, thereby significantly promoting the system's scalability. Thus, the main contributions of this work are as follows:

- We propose a new heuristc named Dynamic Segmentation Strategy (DSS) that allows a flexible segmentation of SFCs. Unlike traditional methods that often rely on static segmentation strategies, DSS enables dynamic adjustments during the placement process. This approach is particularly relevant in multi-domain environments, where network conditions and resource availability can vary considerably within short time frames. The dynamic nature of our segmentation strategy allows the system to quickly adapt to these changes, facilitating better SFC placement and resource utilization.
- We developed a new system model for the SFC segmentation problem. To the best of our knowledge, this is the first model dedicated exclusively to this problem. The model addresses the segmentation in multi-domain environments and introduces elements such as computing and network resources, connectivity, and latency constraints.
- We adapted the combinatorial method "stars and bars" to the SFC segmentation problem. This adaptation provides a systematic way to explore all valid segmentation plans while ensuring the correct order of VNFs. This combinatorial adaptation reduces computational overhead during segmentation once only valid segmentation plans are generated.

This paper is organized as follows. In Section II, we describe the SFC segmentation problem. In Section III, we discuss related work. Section IV presents the proposed system model. Our proposed heuristic named Distributed Segmentation Strategy (DSS) is described in Section V. The experimental results and discussion are shown in Section VI. Finally, Section VII concludes this work and highlights future perspectives.

II. Background

To execute an SFC in a multi-domain environment, the chain of VNFs must be divided into segments. The SFC segmentation problem refers to the challenge of dividing the SFC into smaller, manageable segments that optimize the deployment and operation of VNFs across different domains [7]. This segmentation process aims to balance resource utilization and ensure efficient coordination between domains. The main objectives of SFC segmentation are:

- **Resource Optimization:** Efficiently allocate computational and networking resources for each segment, ensuring that the VNFs operate within the constraints of their respective domains [11].
- **Improved Manageability:** Smaller segments are easier to manage and orchestrate, especially in complex, multi-domain environments. This simplifies the overall management and monitoring of the SFC [1].
- **Enhanced Security and Compliance:** SFC segmentation enables the isolation of specific VNFs to meet security and compliance requirements. This isolation ensures that sensitive data flows through a secured path with the necessary inspection and enforcement points. By using secure tunneling technologies such as VPNs or VXLANs to interconnect segments in different network domains its possible to maintaining security and performance [9].

The SFCs can be executed in a multi-domain environment. A domain is a logical or physical administrative boundary within a network, where resources such as compute nodes, storage, and network infrastructure are managed under a specific control policy [4], [12]. Domains may represent different geographic locations, administrative regions, or service providers, each with its own constraints, resources, and performance characteristics. In multi-domain environments, VNFs within an SFC may be placed across multiple domains, requiring coordination and adherence to the policies and constraints of each domain to ensure optimal service delivery.

The SFC segmentation problem can be mapped to a combinatorial mathematics problem. In this problem, an SFC can be viewed as a set of VNFs and links between them. The goal of the SFC segmentation problem is to partition this set into subsets that are the segments while preserving certain properties, such as order or connectivity. As defined in [13], a valid segmentation plan must meet the following criteria:

1) Each segment must contain at least one VNF.
2) Each VNF should be allocated in exactly one segment.
3) The VNF chain order in the SFC must be maintained inside each segment.
4) The order of segments must follow the VNF chain defined in the SFC.

To identify the number of subsets of a set, it is possible to use the Bell number, denoted as B_n, which indicates the number of possible partitions of a set with n members into non-empty subsets [14]. The recursive formula for the Bell numbers is given by Equation (1).

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \tag{1}$$

Where $\binom{n}{k}$ is the binomial coefficient, and $B_0 = 1$. The use of Bell numbers to compute the total number of segmentation plans cannot be applied directly because some generated subsets do not comply with criteria 3 and 4. For example, suppose that an SFC with VNFs $\{\{v1, v2, v3\}\}$. One of the subsets of B_3 is $\{\{v2, v3\}, \{v1\}\}$ where the order of VNFs within the segments is different from the original SFC, which makes this segmentation plan invalid.

Identifying only valid segments is a relevant step in solving the SFC segmentation problem. Finding all valid segment plans for an SFC is related to the "stars and bars" theorem presented in [10]. It provides a way to determine how to place n indistinguishable objects (stars) into k distinguishable bins (bars). The number of ways to do this is given by the binomial coefficient $\binom{n+k-1}{k-1}$ where n is the number of objects to place and k is the number of bins. The formula computes the number of non-negative integer solutions to equation $x_1+x_2+...+x_k = n_k$, where each $x_i \geq 0$ represents the number of stars in the $i-th$ bin.

In the context of the SFC segmentation problem, the stars represent the VNFs, and the bars represent the boundaries between the segments. The goal is to divide the SFC into multiple segments while preserving the order of VNFs. However, unlike the classical "stars and bars" problem, our approach introduces additional constraints to maintain the sequential order of VNFs within the SFC.

Using all subsets computed based on Bell numbers and removing invalid segments based on the modified "stars and bars" approach, the number of valid segment plans for an SFC with n VNFs is equal to 2^{n-1}. Thus, an SFC composed of 3 VNFs can be segmented into 4 possible valid segmentation plans, as shown in Fig 1.

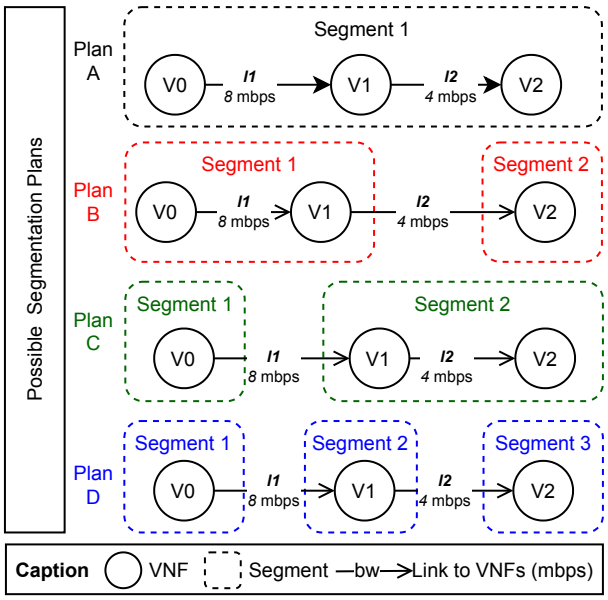

Fig. 1. Valid Segmentation Plans for an SFC with 3 VNFs

As the number of possible segments of an SFC grows exponentially, decreasing the sample space can improve the effectiveness of the placement method. In the following section, we will examine how various placement methods approach SFC segmentation, discussing related work.

III. Related Work

In this section, we discuss how various SFC placement methods address the SFC segmentation problem. Regardless of the chosen SFC placement strategy, segmenting the SFC is a necessary step to allow its execution of SFCs in a multi-domain environment.

In [15], the SFC segmentation problem is solved by developing a decentralized framework that assigns independent Double Deep Q-Learning (DDQL) agents to each Virtualized Infrastructure Manager (VIM). Those agents make local decisions based on detailed observations and minimal exchange of critical information with neighboring agents. That approach improves scalability and resource allocation efficiency. An action coordination module at the NFVO layer aggregates those local decisions into a globally optimized solution.

The authors of [16] address the SFC segmentation problem by proposing a Genetic Algorithm (GA) based solution that dynamically adjusts parameters to adapt to various network topologies and SFC requests, ensuring efficient resource utilization and minimizing segmentation. It uses heuristics for initial population generation to guide the search effectively, optimizing the Cost-to-Revenue Ratio (CRR) for better resource efficiency. The GA-PAGA variant reduces inter-rack traffic by colocating VNFs within the same rack, which lowers latency and bandwidth usage. In general, the solution balances computational load and scalability, making it suitable for large-scale network applications.

In [13], the partitioning phase of the GDM (General Distributed Method) for cross-domain SFC placement involves several key steps to efficiently divide the SFC into segments and assign them to suitable domains. Initially, the receiver domain traverses the SFC, creating segments based on VNFs location constraints. Domains then bid for these segments using a decentralized auction mechanism, calculating bid values with a utility function tailored to their placement goals. Through a consensus-based segment auction (CSA), domains iteratively exchange and update local winning bid lists until a stable segment assignment is achieved. Following this, inter-domain paths are selected to connect the segments, ensuring minimal additional costs and meeting bandwidth requirements. If any segment cannot be assigned due to resource constraints, the algorithm re-divides and re-assigns the segments until all possible placement solutions are considered, thereby maximizing the acceptance ratio and ensuring efficient resource utilization across the substrate network.

The studies cited previously show that the placement of segments in a multi-domain environment is a complex and time-consuming task. Thus, adopting a strategy to choose a suitable segmentation plan with respect to the environment is necessary to reduce the time to place the requested SFC [13]. In the literature, segments are typically defined during the initialization of the placement process and do not change during the placement phase. In contrast, our approach described in Section V, defines segments multiple times during the SFC placement process. This approach of performing multiple segmentations during the SFC placement process offers several advantages i) allowing continuous adjustments that adapt to dynamic changes in network conditions, ii) minimizing the need of sharing information between nodes in the network, and iii) enabling the execution of the placement of the VNFs in the segments in parallel.

IV. System Model

In this section, we describe the system model for our method for SFC segmentation in edge-cloud environments. The model defines the environment and the elements where our proposed heuristic is executed to solve the SFC segmentation problem. The SFC segmentation is performed in edge-cloud nodes. Each node belongs to a single domain. Table I summarizes our symbols for our system model.

TABLE I
SYSTEM MODEL SYMBOLS

Symbol	Description
Parameter Variables	
F	Set of VNFs in the SFC
S	Set of segments in the SFC
D	Set of domains
H_d	Set of nodes in domain d
$R_h = (c_h, m_h)$	Resource capacity of node h
c_h	CPU capacity of node h
m_h	Memory capacity of node h
cpu_f	CPU requirements for VNF f
mem_f	Memory requirements for VNF f
$\text{link}_{f,f'}$	Indicator of connectivity between VNFs
$\text{link\_cap}_{h,h'}$	Bandwidth capacity of the link between nodes h and h'
latency_s	Latency of segment s
latency_threshold	Maximum allowable latency for the SFC
Decision Variables	
$x_{f,s}$	Binary variable: 1 if VNF f is in segment s, 0 otherwise
$y_{s,d}$	Binary variable: 1 if segment s is placed in domain d, 0 otherwise

F is a set of VNFs. Each VNF represents a network function that is part of the SFC. These functions need to be executed in a specific order to provide the desired network service. The cpu_f and mem_f represent the CPU and memory requirements for VNF f. This represents the resource load of a VNF, which must be accommodated within the available node resources.

D is a set of domains. Domains are administrative groupings of network resources that can execute VNFs. The challenge is to allocate VNFs across multiple domains while maintaining service requirements and improving resource use.

S is a set of segments resulting from the SFC segmentation. The segmentation process aims to divide the SFC into manageable parts that can be optimally distributed across domains.

H_d is a set of nodes in domain d. The nodes within a domain provide the computational resources needed to execute VNFs. The allocation of VNFs to nodes must consider resource constraints and the topology of the network. $R_h = (c_h, m_h)$ is the resource capacity of node h. Each node has a limited amount of resources, named CPU (c_h) and memory (m_h). The placement of VNFs must ensure that their total resource requirements do not exceed the capacity of the node.

The $\text{link}_{f,f'}$ is the connectivity indicator between VNFs f and f'. It ensures that VNFs within the same segment can communicate efficiently, following the required order of execution. The $\text{link\_cap}_{h,h'}$ defines the bandwidth capacity of the link. It limits the amount of data that can be transmitted between nodes, influencing the placement of VNFs to minimize internode communication delays. The latency_s defines the delay by processing VNFs within the segment. The latency_threshold acts as a constraint to ensure that the total latency across all segments does not exceed a specified limit of the SFC, thus maintaining the quality of service.

There are two decision variables in the system $x_{f,s}$ and $y_{s,d}$. The first, depicted in Equation (2), defines whether the VNF f is within the segment s. The second, depicted in Equation (3), indicates if the segment s is sent to domain d.

$$x_{f,s} = \begin{cases} 1, & \text{if VNF } f \text{ is included in segment } s \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$y_{s,d} = \begin{cases} 1, & \text{if segment } s \text{ is placed in domain } d \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The problem is subject to some constraints. Equation (4) and Equation (5) ensure that, for each segment s placed in the domain d, the total resources required by the VNFs in the segment must not exceed the resources available in domain d.

$$\sum_{f \in F} x_{f,s} \cdot \text{cpu}_f \leq \sum_{h \in H_d} y_{s,d} \cdot c_h, \quad \forall s \in S, \forall d \in D \tag{4}$$

$$\sum_{f \in F} x_{f,s} \cdot \text{mem}_f \leq \sum_{h \in H_d} y_{s,d} \cdot m_h, \quad \forall s \in S, \forall d \in D \tag{5}$$

Equation (6) imposes the order preservation constraint of the VNFs. This constraint ensures that the sequence in which the VNFs are processed respects the predefined order in the SFC. This means that if a function f_1 precedes the function f_2 in the SFC, this order must be maintained across the segments. This constraint prohibits the creation of segmentation plans where the order of VNFs in the segments differs from their order in the SFC. For example, a plan like $p_1 = (\{v_1, v_3\}, \{v_2\})$ is not allowed for an SFC $(\{v_1, v_2, v_3\})$.

$$x_{f_1,s} \cdot x_{f_2,s} \leq x_{f_1,s} \cdot \sum_{s' \geq s} x_{f_2,s'}, \quad \forall f_1, f_2 \in F, f_1 \to f_2 \tag{6}$$

Equation (7) imposes the connectivity constraints. This constraint ensures that the connectivity requirements between VNFs within the same segment do not exceed the available link capacity between nodes.

$$\sum_{f,f' \in F} x_{f,s} \cdot x_{f',s} \cdot \text{link}_{f,f'} \leq \text{link\_cap}_{h,h'}, \forall h, h' \in H_d \tag{7}$$

Finally, Equation (8) defines the latency constraints. It ensures that the total latency of the segments does not exceed the maximum allowable latency of the SFC.

$$\sum_{s \in S} y_{s,d} \cdot \text{latency}_s \leq \text{latency\_threshold}, \quad \forall d \in D \tag{8}$$

The formulation presented in this section can be modeled as a Constraint Satisfaction Problem (CSP) [17], where the variables represent VNFs, segments, and the constraints include resource limitations, latency requirements, and VNF ordering. The system model provides a comprehensive overview of the SFC segmentation problem in multi-domain environments, defining the key elements and constraints necessary for the association of VNFs across different segments and domains.

By incorporating resource, order preservation, connectivity, and latency constraints, the CSP formulation allows the problem to adapt to different objectives, such as optimizing placement success or minimizing deployment costs. In the next section, we present our method for solving the SFC segmentation problem.

V. Distributed Segmentation Strategy (DSS)

This section presents our heuristic, named Distributed Segmentation Strategy (DSS), for solving the SFC segmentation problem. DSS is modular and decouples the segmentation and placement processes, making it adaptable to various SFC placement methods that operate in multi-domain environments. The proposed segmentation heuristic can be executed concurrently in multiple selected domains to execute the segment placement, encompasses the distributed nature of our approach.

Our proposed heuristic, addresses the SFC segmentation problem by dynamically dividing SFCs into manageable segments for deployment in multi-domain environments. The method iterates through possible segmentation plans, selecting split points in the SFC to create segments that adhere to the VNF order constraint. The heuristic generates valid segmentation plans and sorts them based on the placement strategy to achieve optimal SFC deployment.

The objective during the segmentation phase can vary depending on the requirements of the SFC placement method. For example, if the goal of SFC placement is to reduce the bandwidth consumption between domains, then the segmentation plan can be designed to concentrate the VNFs into a few domains. If the objective of the SFC placement is to maximize the placement success rate, the number of segments can be increased to use resources from different domains. The DSS approach can be configured to meet various goals according to the needs of the SFC placement strategy.

Algorithm 1 Creation of link splitting map

Input: n, the number of VNFs.
Output: Array where each position is as a point of slip.
$valid\_plan\_num \leftarrow 2^{(n-1)}$
$array\_length \leftarrow \log_2(n+1)$
$links\_to\_segment \leftarrow Array(valid\_plan\_num)$
for $i \leftarrow 0$ to $valid\_plan\_num - 1$ **do**
  $bin\_rep \leftarrow intToBinaryString(i)$
  $boolean\_array \leftarrow Array(array\_length)$
  $j \leftarrow 0$
  for each $char$ in bin_rep **do**
    if $char$ == '1' **then**
      $boolean\_array[j] \leftarrow True$
    else
      $boolean\_array[j] \leftarrow False$
    end if
    $j \leftarrow j + 1$
  end for
  $links\_to\_segment[i] \leftarrow boolean\_array\ i \leftarrow i + 1$
end for
return $links\_to\_segment$

Algorithm 1 generates all possible split points for the VNFs of an SFC. The split point is the link that divides the SFC into two segments. Our proposed algorithm creates an array where each element is a boolean array that indicates in each element if a link must be used as the split point or not. This boolean array is used to create the segments via Algorithm 2.

Algorithm 2 groups the VNFs (from an SFC or a segment) into distinct segments. This segmentation is based on a boolean array that specifies where the segments should be split. The goal is to create a segmentation plan that divides the VNFs into segments for further processing or deployment.

Fig 2 depicts the generation of all possible segmentation plans of an SFC with 3 VNFs using Algorithm 1 and Algorithm 2. As we can see in the figure, each possible segmentation plan indicates which links are used to split the segments. In the first segmentation plan, both link $L1$ and $L2$ are not chosen as split points, resulting in a segmentation plan with a single segment, $s_1 = \{v_0, v_1, v_2\}$ containing all the VNFs. In contrast, in the last segmentation plan, links $L1$ and $L2$ are selected as split points, producing 3 segments $s_1 = \{v_0\}, s_2 = \{v_1\}, s_3 = \{v_2\}$, with only one VNF.

Algorithm 2 Creation of segmentation plans

Input:
$vnfs$: Array with the VNFs of the SFC
$links\_to\_segment$: Array with booleans
Output: List of segments.
$segmentation\_plan \leftarrow List()$
$i \leftarrow 0$
while $i <$ *size*(vnfs) **do**
  $segment \leftarrow List()$
  $segment.insert(vnfs[i])$
  $current\_index \leftarrow i$
  for $j \leftarrow i$ to *size*($links\_to\_segment$) - 1 **do**
    if $links\_to\_segment[j]$ == *True* **then**
      break
    end if
    $segment.insert(vnfs[j+1])$
    $current\_index \leftarrow j + 1$
  end for
  $i \leftarrow current\_index$
  $segmentation\_plan.insert(segment)$
end while
return $segmentation\_plan$

Fig 3 depicts our proposed heuristic. The first parameter is a list of VNFs, this list can be all the VNFs of an SFC, or a portion of a previously divided SFC. The second parameter is the sort strategy employed by the placement algorithm to achieve its objectives.

The first step consists of using Algorithm 1 to create the link splitting map. The number of VNFs requested is used as a parameter. The result of Algorithm 1 is an array with at least one element.

The next step consists of iterating over this array and, for each element, Algorithm 2 is executed. The result of

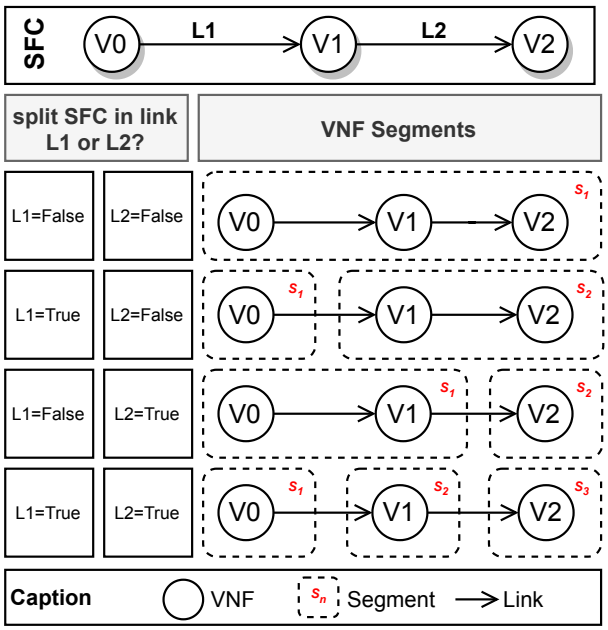

Fig. 2. Example of segmentation with 3 VNFs

Algorithm 2 is the segmentation plan based on the link splitting map. All the possible segmentation plans are stored in list ***S***.

After creating all possible segmentation plans, they are sorted based on the sorting strategy provided by the placement algorithm. This sorting strategy is a method used to rank the segmentation plans created. After generating all valid segmentation plans, the sorting strategy sorts them based on specific criteria defined by the SFC placement method, such as minimizing resource usage, reducing latency, or optimizing placement success. This ensures that the most suitable segmentation plan, aligned with the objectives of the system, is tested first during the placement phase. Thus, the first element of the list is returned to the SFC placement component as the select segmentation plan.

The subsequent step involves examining the newly created segments to determine if they have a candidate domain to be placed or if they need to be resegmented. Various strategies can be employed to verify whether the segment can be placed or not, and each placement component may approach this differently, such as using MIP, Game Theory, Auction, Full Mesh Aggregation, etc. If a created segment lacks a candidate domain capable of handling all its VNFs, the segment is reintroduced into the algorithm for resegmentation.

Segments with candidate domains for execution are handed over to the placement component. Once the segment reaches the chosen domain, the segmentation process can be performed once more, if necessary. This process persists until all requested VNFs are allocated to a domain capable of providing the necessary resources for their execution. Our proposed heuristic allows different segments to be segmented and placed in different domains simultaneously, reducing the overall time required for SFC placement.

In some cases, the resources available within the domains are insufficient to execute the SFC. The heuristics we propose stop when the segments created have a single VNF and there is no domain available to execute the VNF of the segment. In this case, the SFC placement fails.

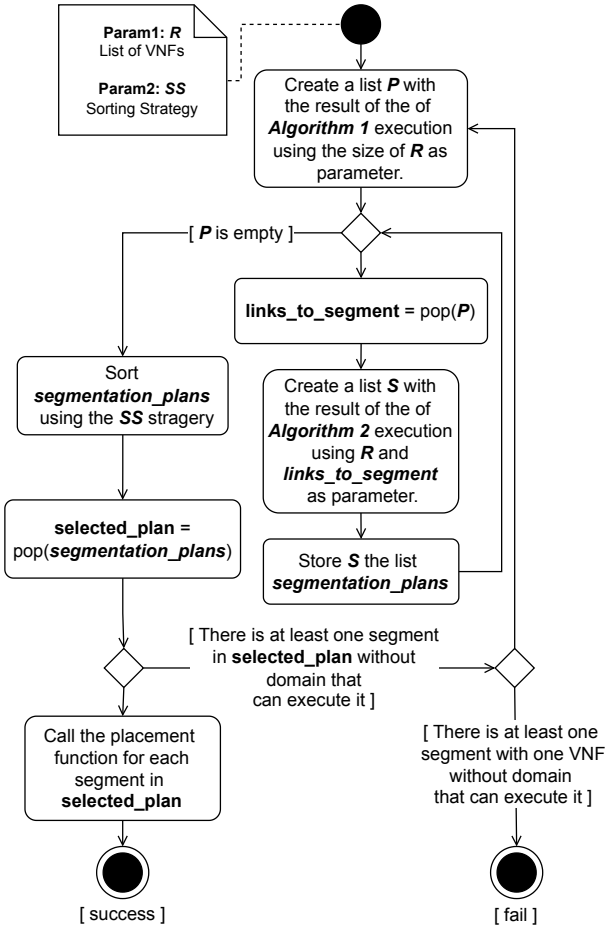

Fig. 3. Segmentation Heuristic

The process of generating all segmentation plans involves creating subsets of VNFs, which corresponds to the "stars and bars" problem. For an SFC with n VNFs, the number of valid segmentation plans grows exponentially, almost $O(2^{(n-1)})$, where n represents the number of VNFs. This means that the time required to create all segmentation plans varies depending on the number of VNFs in the SFC.

VI. Performance Evaluation

This section presents the results of our performance evaluation for DSS in different SFC placement scenarios. The experiments were conducted on a system with an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz. The system is equipped with 4 physical cores (8 threads) and a cache size of 8MB. The total available memory is 20GB. Our proposed heuristic solves the SFC segmentation problem in the Edge-Cloud continuum and can be used with different SFC placement methods. This unique characteristic of our proposed heuristic

TABLE II
SIMULATION PARAMETER SETTINGS

Description	Values
Number of Domains	64
Compute Nodes in Domain	[5,10]
CPU capacity	[1,5] GHz
Memory capacity	[1,5] GB
Bandwidth capacity in inter-domain link	1000 Mbps
Bandwidth capacity in intra-domain link	2000 Mbps
Link Delay	[2, 5] ms
CPU Cost	[0.001, 0.005] $/s
Memory Cost	[0.001, 0.004] $/GBs
SFC Size (Number of VNFs)	[3 - 6]
CPU demand of VNF	[1,4] GHz
Memory demand of VNF	[1,5] GB
Traffic rate requirement	[100, 500] kbps
Traffic routing cost parameter	[0.02, 0.05] $/Mb
Maximum tolerated delay	[0, 50] ms

imposes challenges for a fair comparison with the solutions already proposed for the SFC segmentation problem. Therefore, we conduct a series of experiments that evaluate the time necessary to build the segmentation plan and the success rate of the SFC placement with different SFC placement configurations.

Table II shows the simulation parameters, which were defined on the basis of [18]. We use Internodes backbone topology, from the Internet Topology Zoo [19]. Each of the 64 nodes in the topology was considered a domain. We create scenarios with 5, 10, 20, 50, and 100 SFC requests. The SFC requests arrive in an interval following a Poisson distribution. The VNF Types in the SFC represent network services like firewalls, intrusion detection, cache, flow monitor, load balance, and Wan-opt. The minimum CPU and memory requirements for each VNF Type were obtained from the respective developer's websites. The simulation was implemented using SimPy [20]. The simulations were executed 10 times for each scenario. The confidence level of the results is 95%. We adopt the SFC with 3 - 6 VNFs, equivalent to the value presented in [21].

Solving the SFC placement Problem takes considerable time [3]. Therefore, the segmentation process must be time efficient to ensure that it does not compromise the overall duration of SFC placement. Fig 4 illustrates the average time required to create the segmentation plan as the number of VNFs in the SFC changes. The x-axis indicates the number of VNFs in the SFC, while the y-axis represents the average time it takes to generate the segmentation plan, and the y-axis is in logarithmic scale. We compared the DSS approach with a naive method that generates all possible combinations of VNFs. The results show that by focusing exclusively on the generation of valid plans, the DSS approach reduces the overall generation time.

Fig 5 shows the success rates of SFC placement using a placement method based on Singleton Congestion Game [22], which is a type of game in Game Theory where each player selects a single resource from a set, and the cost or utility for using that resource depends on how many players choose the same resource, thus, players aim to minimize their individual costs, which increase as more players select the same resource,

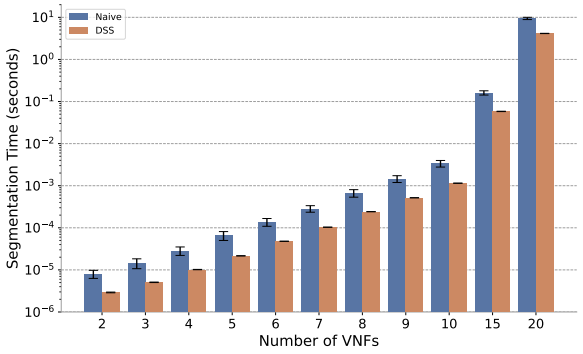

Fig. 4. Segmentation Time by SFC Size

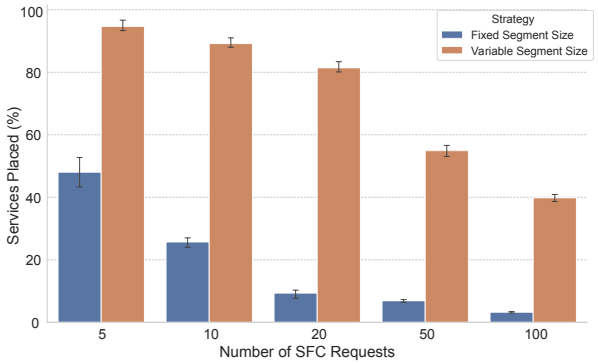

Fig. 5. Impact of Segment Size on SFC Placement Success Rates

creating "congestion" on that resource. The implementation was configured using our proposed heuristic with two segmentation strategies i) Fixed Segment Size and ii) Variable Segment Size. The x-axis represents the number of SFC requests [5, 10, 20, 50, 100], while the y-axis shows the percentage of SFC placed successfully. The results indicate that the variable segment size strategy achieves higher placement success rates, especially as the number of requests increases. This suggests that allowing segment sizes to adapt dynamically to changing conditions leads to more efficient resource allocation and improved network performance compared to a fixed segment size approach.

Fig 6 presents the execution of DSS in an SFC placement method based on a greedy approach. The Placement was executed using two segmentation strategies, the first one tries to reduce the placement cost selecting the nodes with lower execution cost and the second tries to increase the number of placed SFCs. The x-axis represents the number of SFC requests [5, 10, 20, 50, 100], and the y-axis shows the SFCs successfully placed. The results demonstrate that the cost-reduction strategy tends to reduce the SFC placement success rate, particularly as the number of requests increases.

In summary, we can conclude that our proposed heuristic achieved its objective. Firstly, as shown in Fig 4, the time required to generate the placement plan for an SFC with 20 VNFs is less than 10 ms, thus not negatively impacting the

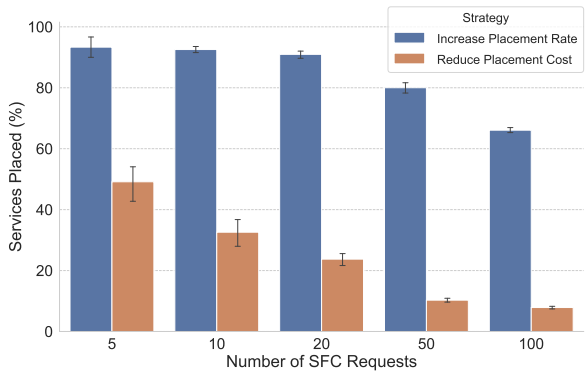

Fig. 6. Impact of Cost Minimization on SFC Placement Success Rates

total SFC placement time. Secondly, we integrate DSS into two distinct SFC placement methods. The results in Fig 5 come from a Game Theory approach, while the results in Fig 6 are based on a Greedy approach. This shows that our approach can be combined with multiple SFC placement methods. Finally, the results in Fig 5 and Fig 6 demonstrate that our approach can serve multiple purposes, including reducing SFC placement costs and increasing the SFC placement rate.

VII. Conclusion

In this work, we proposed a new heuristic to solving the SFC segmentation problem named the Distributed Segmentation Strategy (DSS). Our method can be combined with different SFC placement methods once it decouples segmentation from the placement process. DSS is inspired by the "stars and bars" theorem.

The experimental results indicate that our method is applicable to various SFC placement methods. In addition, the experiments show that the SFC segmentation strategy is an important factor in terms of the success rate of the SFC placement and the cost of SFC execution. Therefore, choosing the appropriate segmentation method allows the entire system to meet the requirements of the infrastructure owner.

In future work, we will implement the proposed heuristic as a component within platforms such as Open Source MANO (OSM) or the Edge Multi-Cluster Orchestrator (EMCO). This implementation will allow evaluations of our approach in real-world scenarios. In addition, we intend to perform more robust comparative analysis with our approach varying the different parameters that determine the requirements of the VNFs and also of the edge-cloud environment topology.

Acknowledgment

This work was supported by CAPES, CNPq, MCTI, CAPES-Print, RNP, OpenRAN@Brasil, and FAPERJ. Flavia C. Delicato and Débora Muchaluat-Saade are CNPq and FAPERJ Fellows.

References

[1] G. Moualla, T. Turletti, and D. Saucez, "An availability-aware sfc placement algorithm for fat-tree data centers," in *CloudNet*, 2018, pp. 1–4.

[2] M. Pattaranantakul, C. Vorakulpipat, and T. Takahashi, "Service function chaining security survey: Addressing security challenges and threats," *Computer Networks*, vol. 221, p. 109484, 2023.

[3] E. L. C. Macedo, A. L. E. Battisti, J. L. Vieira, J. Noce, P. F. Pires, D. C. Muchaluat-Saade, A. C. B. Oliveira, and F. C. Delicato, "Distributed auction-based sfc placement in a multi-domain 5g environment," *SN Computer Science*, vol. 5, no. 1, Dec. 2023.

[4] G. L. Santos, D. d. F. Bezerra, É. d. S. Rocha, L. Ferreira, A. L. C. Moreira, G. E. Gonçalves, M. V. Marquezini, Á. Recse, A. Mehta, J. Kelner, D. Sadok, and P. T. Endo, "Service Function Chain Placement in Distributed Scenarios: A Systematic Review," *Journal of Network and Systems Management*, vol. 30, no. 1, 2022.

[5] A. L. É. Battisti, E. L. C. Macedo, M. I. P. Josué, H. Barbalho, F. C. Delicato, D. C. Muchaluat-Saade, P. F. Pires, D. P. d. Mattos, and A. C. B. d. Oliveira, "A novel strategy for vnf placement in edge computing environments," *Future Internet*, vol. 14, no. 12, p. 361, 2022.

[6] D. O. Rodrigues, A. M. De Souza, T. Braun, G. Maia, A. A. F. Loureiro, and L. A. Villas, "Service provisioning in edge-cloud continuum: Emerging applications for mobile devices," *Journal of Internet Services and Applications*, vol. 14, no. 1, p. 47–83, Jun. 2023.

[7] A. Abujoda and P. Papadimitriou, "Distnse: Distributed network service embedding across multiple providers," in *COMSNETS*, 2016, pp. 1–8.

[8] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A link failure recovery algorithm for Virtual Network Function chaining," *Proceedings of the IM 2017 - 2017 IFIP/IEEE*, no. May, 2017.

[9] A. Huff, G. Venâncio, V. F. Garcia, and E. P. Duarte, "Building multi-domain service function chains based on multiple nfv orchestrators," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2020, pp. 19–24.

[10] R. P. Stanley, "Enumerative combinatorics volume 1 second edition," *Cambridge studies in advanced mathematics*, 2011.

[11] Y. Xu and V. P. Kafle, "Optimal service function chain placement modeling for minimizing setup and operation cost," in *IEEE CloudNet*, 2018, pp. 1–3.

[12] R. Viola, A. Martín, M. Zorrilla, J. Montalbán, P. Angueira, and G.-M. Muntean, "A survey on virtual network functions for media streaming: Solutions and future challenges," *ACM Computing Surveys*, vol. 55, no. 11, 2023.

[13] Y. Liu, H. Zhang, D. Chang, and H. Hu, "GDM: A General Distributed Method for Cross-Domain Service Function Chain Embedding," *IEEE TNSM*, vol. 17, no. 3, 2020.

[14] M. Z. Spivey, "A generalized recurrence for bell numbers," *J. Integer Seq*, vol. 11, no. 08.2, p. 5, 2008.

[15] A. Pentelas, D. De Vleeschauwer, C.-Y. Chang, K. De Schepper, and P. Papadimitriou, "Deep multi-agent reinforcement learning with minimal cross-agent communication for sfc partitioning," *IEEE Access*, vol. 11, pp. 40 384–40 398, 2023.

[16] P. Rodis and P. Papadimitriou, "Intelligent and resource-conserving service function chain (sfc) embedding," *Journal of Network and Systems Management*, vol. 31, no. 4, Sep. 2023.

[17] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European journal of operational research*, vol. 119, no. 3, pp. 557–581, 1999.

[18] C. Chen, L. Nagel, L. Cui, and F. P. Tso, "Distributed federated service chaining: A scalable and cost-aware approach for multi-domain networks," *Computer Networks*, vol. 212, p. 109044, Jul. 2022.

[19] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, 2011.

[20] SimPy Development Team, *SimPy: Discrete Event Simulation for Python*, n.d., available at https://simpy.readthedocs.io/.

[21] Y. Li, L. Li, J. Bai, X. Chang, Y. Yao, and P. Liu, "Availability and reliability of service function chain: A quantitative evaluation view," *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, Apr. 2023.

[22] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, 1973.