Breaking the Limits: Bio-Inspired SFC Deployment across Multiple Domains, Clouds and Orchestrators

Vinicius Fulber-Garcia*, José Flauzino*, Giovanni Venâncio*, Alexandre Huff†, Elias P. Duarte Junior*
*Federal University of Paraná, UFPR, Curitiba, Brazil
†Federal Technological University of Paraná, UTFPR, Toledo, Brazil
Email: {vinicius, jwvflauzino, gvsouza, elias}@inf.ufpr.br, alexandrehuff@utfpr.edu.br

***Abstract*—The Multi-SFC strategy enables the deployment of virtual network services across multiple clouds, domains, and NFV orchestrators. In this context, finding optimal solutions to the NFV Resource Allocation (NFV-RA) problem, and in particular the VNF Forwarding Graph Embedding (VNF-FGE) problem, becomes a formidable challenge. This translates into a multi-objective optimization problem that involves multiple heterogeneous resources, services and functions, each with particular sets of features and restrictions. In this work, we propose a bio-inspired strategy for mapping Multi-SFCs, which consists of a multi-objective genetic algorithm that minimizes inter-domain latency and resource usage costs, while maximizing inter-domain bandwidth availability. Furthermore, the solution is dynamic in the sense that service re-mapping can be invoked after network conditions change. Simulation results show the effectiveness of the proposal to create and adapt optimized solutions.**

***Index Terms*—NFV; SFC; Deployment; Mapping; Genetic.**

I. Introduction

Network Functions Virtualization (NFV) allows the implementation of network functions, traditionally deployed on specialized hardware, in software [1]. Each instance of a Virtualized Network Function (VNF) is executed using virtualization technologies such as full virtualization, paravirtualization, and containerization [2]. The NFV reference architecture proposed by the European Telecommunications Standards Institute (ETSI) has emerged as the main standard for the execution of NFV functions and services and also the management and orchestration of VNFs [3]. The reference architecture also enables the composition of Service Function Chains (SFCs) to build network services. An SFC consists of multiple network functions connected in a predefined order to process network traffic [4].

Typically, network functions of an SFC are instantiated at the same Point of Presence (PoP), such as a cloud or a domain. However, this approach is limited by several factors, including the availability of resources in a cloud [5] or even functions that operate natively in specific domains. In addition, existing NFV management and orchestration platforms (*e.g.*, OpenStack Tacker [6] and Open Source MANO [7]) provide homogeneous orchestration modules – although these platforms support the coordination of multiple orchestrators distributed across different points of presence, they do not allow cooperation between different orchestration modules. However, with the increasing complexity of virtualized services and the variety of NFV execution, management, and orchestration solutions [8]–[10], the distribution of SFCs across multiple domains and their orchestration and execution on heterogeneous NFV platforms becomes a natural requirement.

In this context, the Multi-SFC [11] has emerged as a solution that enables SFCs to be distributed, executed, and managed across multiple clouds, domains, and NFV orchestrators. The Multi-SFC considers that an SFC comprises multiple segments that are instantiated at different PoPs. A segment represents the fundamental building block of the Multi-SFC and is composed of a sequence of VNFs. Each segment is allocated to a specific domain and then connected to other segments running on different clouds/domains to build complex end-to-end services.

One of the main challenges of the Multi-SFC is the solution of the VNF Forwarding Graph Embedding (VNF-FGE) problem, which is part of the broader NFV Resource Allocation (NFV-RA) problem [2], [12]. This optimization problem consists of deploying SFCs in infrastructures with heterogeneous resources. In particular, it requires optimization that considers multiple objectives and constraints to allocate services, segments, and functions.

In this work, we propose the Service Mapping Expedient for Networked Traffic and Environments (SeMENTE): a bio-inspired strategy for dynamic allocation of virtualized network services in the context of Multi-SFC. The SeMENTE solution defines a multi-objective genetic algorithm that allows specifying and mapping segments of a given SFC aiming to optimize resource allocation across different clouds, domains, and NFV orchestrators. The primary goals of SeMENTE consists of minimizing inter-domain latency and financial costs associated with computing resource usage while maximizing inter-domain bandwidth availability. In addition, SeMENTE considers a set of constraints imposed by both the service to be mapped and the infrastructure in which it will be deployed.

Furthermore, in complex production environments such as the ones enabled by the Multi-SFC, it becomes natural for the network state, available resources, and their corresponding costs to alter gradually over time [13]. Therefore, even if a service is optimally deployed, its performance may degrade over time, potentially resulting in higher costs than those that have been expected in a previous mapping process. To overcome that, SeMENTE includes an on-demand SFC *re*mapping feature that considers a scenario with multiple clouds, domains, and NFV orchestrators. To accomplish this feature, our

solution employs historical data from the last service mapping to introduce the best individuals found in a previous mapping as the initial population for a new mapping process. This technique aims to speed up the algorithm convergence while achieving high-quality results. Given the incremental nature of changes in the network, it is expected that the old mapping may still have highly suitable segments for a given service.

Experiments were conducted to evaluate the capability of SeMENTE to find optimized solutions for the mapping problem, considering the average execution time of the solution under different configurations, including different mapping constraints and dependencies. Results show the efficiency and effectiveness of SeMENTE operating successfully in complex mapping scenarios. It creates optimized solutions to the dynamic mapping problem. Furthermore, the running times grow linearly with the size of the problem. In particular, we highlight the superior performance of the solution for remapping the virtual services after significant changes of the quality of service.

The rest of this paper is organized as follows. Section II presents an overview of the Multi-SFC. The proposed solution is described in Section III, including the design and implementation. Section IV presents the experimental evaluation. Related work is in Section V. Finally, the conclusions are in Section VI.

II. Multi-SFC: SFCs Across Multiple Domains, Clouds, and Orchestrators

The Multi-SFC is a strategy that enables the instantiation and management of SFCs across multiple domains, clouds, and NFV orchestrators [11]. In the context of Multi-SFC, an SFC consists of multiple segments, each containing a subset of the VNFs that compose the SFC. A given segment is allocated to a specific domain and is connected to other segments from different domains. Figure 1 illustrates multiple segments of a single service implemented as a Multi-SFC. A segment is allocated to a domain, and communication between segments is tunneled, ensuring security even across the Internet.

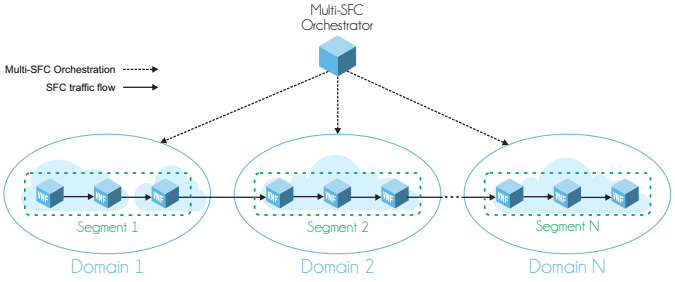

Fig. 1. Multi-SFC: multiple segments of a service.

Tunnels provide an abstraction for communication between pairs of segments. In the Multi-SFC, all tunnels are executed through VNFs implementing, for example, a Virtual Private Network (VPN). Considering the IETF SFC architecture, a VNF that implements a tunnel corresponds to the inter-domain Service Function Forwarder (SFF). Any SFC can be converted into a Multi-SFC by defining the sequence of VNFs from the original SFC to run on different domains and creating a tunnel between each segment. Thus, both the output and input points between a pair of segments are tunneling VNFs. The tunneling VNF-based strategy frees network operators from the myriad of manual configurations required to interconnect different Multi-SFC domains. Figure 2 illustrates a tunnel between two domains.

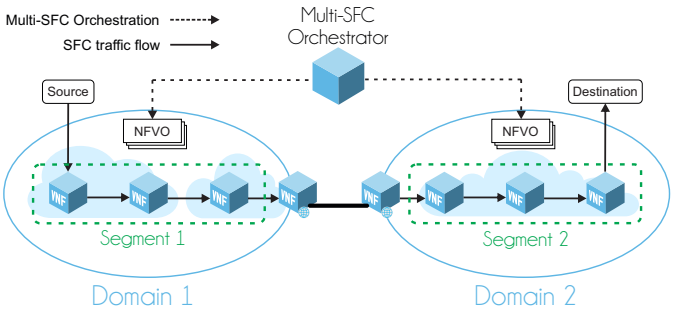

Fig. 2. Multi-SFC interconnecting a pair of domains.

The main module of the Multi-SFC architecture is the Multi-SFC Orchestrator, also shown in Figure 2. This module was proposed according to the definitions of the ETSI NFV-MANO framework to ensure interoperability between different NFV solutions. The Multi-SFC Orchestrator implements a set of functionalities that enable the composition and orchestration of SFCs in each domain. These functionalities include VNF and SFC descriptor management, instantiation, and deletion. The module also translates generic Multi-SFC operations into the corresponding operations of each NFV orchestrator and configures SFCs between different NFV platforms, including the automatic forwarding of inter-domain traffic. The Multi-SFC Orchestrator includes one network agent for each domain. An agent is responsible for automatically and transparently configuring routing and VPN tunnels for each segment to enable traffic forwarding between multiple domains.

The Multi-SFC also provides an Application Programming Interface (API) to enable the integration of multiple client applications. The functionalities abstracted by the Multi-SFC API refer to the operations for instantiating, querying, and destroying SFCs and their VNFs on the different NFV platforms. The Multi-SFC also manages all configurations required to interconnect the SFC segments distributed in different domains.

III. Bio-inspired Multi-domain Mapping in the Context of Multi-SFC

This work proposes SeMENTE, a solution to the NFV Resource Allocation (NFV-RA) problem in the context of the Multi-SFC. In particular, SeMENTE considers the allocation and mapping of network functions for inter-domain tunneling, whenever necessary. SeMENTE also includes a *re*mapping feature that can be invoked whenever the quality of service deviates enough from that required by the original mapping. Furthermore, the proposed mapping strategy verifies the occurrence of the following dependencies that can be defined for

each of the functions composing a service: *(i)* domain specific; *(ii)* domain types (*e.g.*, cloud, fog, edge) and; *(iii)* orchestrator types (*e.g.*, OpenStack Tacker and Open Source MANO).

The SeMENTE solution uses an optimization model based on three main metrics: deployment cost, inter-domain latency, and inter-domain bandwidth. The deployment cost refers to the financial expenses for deploying and maintaining VNFs in specific domains or PoPs. Inter-domain latency refers to the round-trip time observed of specific communication links between different domains or PoPs in the network. Finally, inter-domain bandwidth refers to the data transmission capacity on the links mentioned above. The optimization objective is to minimize the sum of inter-domain cost and latency while maximizing the sum of inter-domain bandwidth. We employed the Pareto frontier analysis model [14].

The service request document model adopted by SeMENTE is an YAML (YAML Ain't Markup Language [15]) document, organized into three main sections: `DOMAINS`, `REQUIREMENTS`, and `TOPOLOGY`. The `DOMAINS` section defines the PoPs available for a network function, specifying characteristics such as cost, type, available orchestrators, and possible transitions between domains, where available bandwidth and latency values are predetermined. The `REQUIREMENTS` section specifies optional relational expressions that determine the acceptance or rejection of a mapping based on bounds defined for the optimization metrics. Finally, the `TOPOLOGY` section uses a modification of the Service ChAin Grammar (SCAG) [16] model to define the service chains, incorporating markers to indicate dependencies of domain types and orchestrators.

The proposed solution was implemented[1] in Python 3 using a modified version of the Platypus multi-objective optimization library [17]. The genetic heuristic chosen was the Non-dominated Sorting Genetic Algorithm II (NSGAII) [18], due to its elitist properties (*i.e.*, the persistence of the best individuals), it is suitable for the mapping problem addressed in this work. The main features of the solution in the context of genetic algorithms are detailed below.

Individual. An individual defines a mapping option for a network service in a particular environment. Each individual has a genotype, represented by a vector, that reflects the requested service chain. Each gene (*i.e.,* position in the vector) in the genotype represents a specific network function of the service, and the allele (*i.e.,* value at a position in the vector) of a gene indicates the PoP at which the corresponding function should be mapped.

Population. A population is the set of individuals at a given time, called a generation. These individuals go through reproductive processes to produce the next generation. Note that the best individuals in the population are preserved in the next generation by using an elitist algorithm – as determined by the relative Pareto frontier.

Initialization. The initial population (first generation) is generated using a generation operator. In the context of SeMENTE, the chosen generator employs a random generation strategy while ensuring all domain dependencies and the presence of valid communication links between domains in case of transitions. The generator can be reapplied during the execution of the genetic algorithm, improving the genetic variability of a generation and the ability of the algorithm to explore the search space.

Selection. This stage selects the individuals to be submitted to the crossover process. The proposed solution adopts a binary tournament selection, where two individuals are randomly selected from the population and the one with the best results for the evaluated metrics is selected for the crossover.

Crossover. The crossover operator plays a central role in generating individuals of the G generation from the $G-1$ generation by merging genes. In SeMENTE, the Simulated Binary Crossover (SBX) [19] crossover algorithm was used to choose a mode of operation that does not produce side effects in maintaining potential domain dependencies.

Mutation. The mutation operator changes alleles of specific genes using the controlled bit swapping technique, where random bits of the allele value – an index indicating a specific PoP or domain – are modified to create a new valid index, thus indicating a new PoP. It is relevant to note that the mutation operator does not act on genes with domain dependencies.

The proposed solution allows the user to configure several parameters to influence its execution and, consequently, the generated results. The main parameters that enable fine-tuning the algorithm are as follows: the criterion to stop the algorithm which can be either a timeout or a maximum number of generations; population size, and; crossover and mutation rate. The execution model of SeMENTE is organized in a cycle of seven (7) stages, as shown in Figure 3.

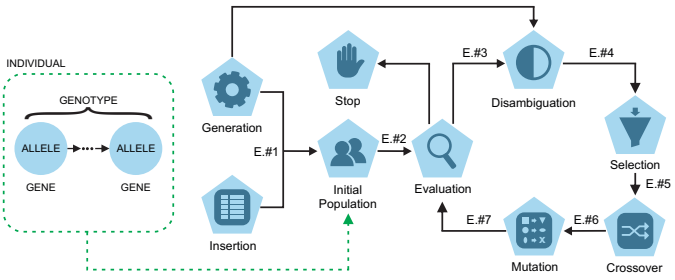

Fig. 3. SeMENTE execution model.

In the first stage (`E.#1`), the initial population is created using either a previously introduced generation mechanism or an insertion mechanism. Unlike the generation mechanism, the insertion mechanism does not generate random individuals to form the initial population; instead, it creates a list of previously selected individuals. However, individuals that were considered highly suitable for the requested mapping at the time X may not fully represent the best solution at the time $X+Y$ due to network changes. On the other hand, it is important to note that changes in the network are usually gradual and not catastrophic. Therefore, certain genes from the best individuals at time X may still be highly suitable at

[1]Available at https://github.com/ViniGarcia/SeMENTE

time $X + Y$. Thus, using these genes in the initial population tends to speed up the algorithm convergence and improve the quality of the newly generated individuals.

The second stage (`E.#2`) evaluates the initial population based on the objective function, which aims to minimize the sum of the financial cost and the total inter-domain latency, while maximizing the sum of the inter-domain bandwidth. This evaluation quantifies the individual suitability to solve the required mapping problem. The third stage (`E.#3`) involves occasional adjustments to the population to improve its variability, balancing processes of exploring and exploiting the solution. At this stage, in every tenth generation, a cleaning process takes place, in which N individuals on the current Pareto frontier with the same fitness value (*i.e.*, the same score for all considered metrics) are reduced to 1. The total number of individuals eliminated in the cleaning process is balanced by the population with the generation of new random individuals, which are evaluated immediately.

Steps `E.#4`, `E.#5` and `E.#6` are responsible for creating the next generation of individuals by sequentially applying the selection, crossover, and mutation operators to the current population. The seventh stage (`E.#7`) consists of the evaluation of the newly formed generation and marks the beginning of a new cycle that steps through stages three (3) to seven (7). This execution cycle is repeated until the stop criteria is reached. Finally, the solution outputs the current Pareto frontier.

IV. Experimental Evaluation

The SeMENTE solution was evaluated through simulation to determine the efficiency and effectiveness of the service mapping strategy in distributed and heterogeneous environments. The tested services consist of SFCs with a linear topology containing from 5 to 10 VNFs. A complete graph with 150 vertices represents the distributed environment. Vertices are PoPs and edges are links between PoPs. A PoP can be located in the cloud, the fog, or at the edge and can support 1 to 4 NFV-MANO platforms. The graph and the values for the optimization metrics related to vertices (financial cost) and edges (latency and bandwidth) were randomly generated within a predefined range of values. Although the topology of real networks are typically not complete graphs, this type of graph actually increases the search space, thus resulting in a more challenging optimization scenario.

The following genetic configurations were used: population size of 100, crossover rate of 100%, and mutation rate of 30%. These configurations have been chosen based on previous runs, which indicated that the mapping problem benefits from populations with high variability that prevented the algorithm from stagnating at local optimal points. The experiments ran on a server with an Intel Core i5-3330 @ 3.0 GHz CPU, 8 GB of RAM, and Ubuntu 20.04. Each experiment was repeated 30 times.

Results have been analyzed considering the evaluation technique for multi-objective Pareto frontier optimizations [20], in particular relative Pareto frontiers. In this technique, individuals on the outermost frontier (frontier 0 or Pareto frontier) are called non-dominated individuals, since they always have better values in at least one of their optimization metrics in comparison with those coexisting on the same frontier. This also applies to the innermost frontiers, with the addition that, there is at least one individual that is the best in all evaluated aspects (dominant) on all outermost frontiers. For example, for every individual on frontier 1, there is at least one dominating individual on frontier 0. The absolute Pareto frontier is determined by having access to all possible solutions to a given problem. Conversely, the relative Pareto frontier is computed by considering all possible solutions found during the optimization process or step, regardless of whether they represent the complete set.

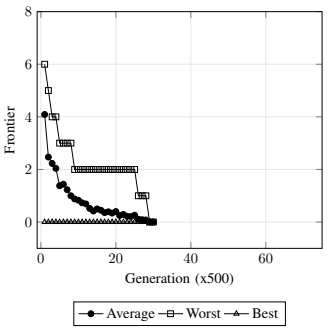

Fig. 4. Convergence (T.5)

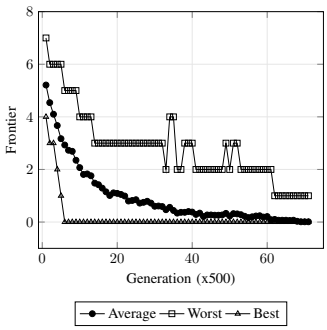

Fig. 5. Convergence (T.10)

The first experiment evaluates the convergence of the algorithm to a relative Pareto frontier. The convergence verification is executed after 500 generations (one execution step), and the absence of changes in the Pareto frontier in an execution step defines the stopping criteria. Figures 4 (5 functions) and 5 (10 functions) show the results obtained step by step from a single execution until reaching the convergence criteria, indicating the best, worst, and average of the frontiers where individuals from the Pareto frontier are present at a given time. Results show the ability of the algorithm to converge to a relative Pareto frontier over time. However, the number of generations required to achieve this convergence is directly proportional to the search space size. Therefore, even if the distributed environment is the same for both test cases, the chain size directly affects the number of possible combinations to generate valid mapping results. For example, 30 execution steps were required to achieve convergence for an SFC with 5 functions, in contrast to the requirement of 71 steps to detect convergence for an SFC with 10 functions.

The second experiment presents results regarding the execution time as the number of generations varies. Figures 6 and 7 show the average and standard deviation of the execution time for an SFC containing 5 and 10 VNFs, respectively. It is possible to observe a linear progression of the execution time, indicating that when the number of generations to be executed doubles, the execution time also approximately doubles. Another relevant point is that there is no significant change in execution time for the same number of generations, regardless of the size of the service chain. This phenomenon occurs

because chain evaluation consists of arithmetic processes with low computational cost compared to genetic routines and Pareto frontier analysis. Consequently, the chain size is not a determinant factor for the execution time given a certain number of generations. However, larger SFCs require higher generation numbers to converge, typically resulting in longer execution times to achieve globally optimal results.

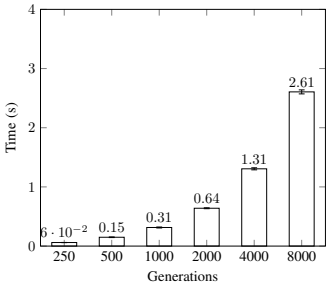

Fig. 6. Time (T.5)

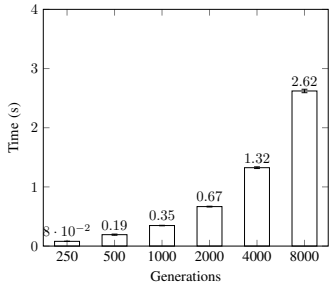

Fig. 7. Time (T.10)

The third experiment evaluates different service chain mappings. The results shown in Table I highlight the best candidates for the mapping (individuals on the relative Pareto frontier), considering each optimization metric. For example, the curve for Cost shows the candidate with the lowest cost found on the relative Pareto frontier, and in brackets, the `(Cost, Latency, Bandwidth)` for the candidate. In this experiment, the stopping criteria adopted was a time limit that allowed the genetic algorithm to run for 60 seconds. Three different cases were considered: *(i)* `Free`, where no constraint or dependency was defined for the mapping of the SFCs; *(ii)* `Orchestrator Defined`, where all functions depend on the availability of the OpenStack Tacker orchestrator at the selected PoP; and *(iii)* `Zero Latency`, where a zero latency constraint is applied to the mapping.

TABLE I
THE BEST RESULTS FOR EACH OPTIMIZATION METRIC.

		Cost ($)	Latency (ms)	Bandwidth (Mbps)
T.5	**Free**	569 (569, 237, 46524)	30 (921, 30, 44601)	158094 (3714, 215, 158094)
	Orchestrator Defined	611 (611, 158, 9716)	17 (1127, 17, 17151)	152012 (4747, 330, 152012)
	Zero Latency	-	-	-
T.10	**Free**	1914 (1914, 883, 199424)	168 (14319, 168, 183098)	344252 (11820, 780, 344252)
	Orchestrator Defined	3607 (3607, 580, 224508)	221 (7032, 221, 141849)	340813 (8220, 962, 340813)
	Zero Latency	-	-	-

The `Free` case produced optimal results for all metrics. This case represents a baseline, as it uses the largest search space, since all PoPs are available in the search for candidate mappings. The `Orchestrator Defined` case restricts the search space, since only PoPs that support OpenStack Tacker can be used for mapping. This dependency, which applies to all functions, limits the choice of PoPs from 150 to 70. Satisfying this domain dependency has two immediate effects: *(i)* the elimination of good mapping candidates as they use domains without the presence of the required orchestrator, and; *(ii)* the best mapping alternatives are found faster, since only a single search subspace for individuals is explored throughout the execution of the solution, compared to the search space of the `Free` case.

The `Zero latency` case represents a scenario in which SeMENTE cannot find viable candidates to solve the optimization problem. In this case, the network service functions have no dependencies, but a restriction is imposed on the optimization process: the inter-domain latency must be zero. This restriction implies that all network functions of the service must be mapped to the same PoP (domain), which limits the number of possible valid solutions from 150^n (where n is the number of network functions in the SFC) to 150. Thus, considering that the variability of the population is improved by mixing and modifying its genetic characteristics, coupled with an extremely limited search space, the process of generating new individuals by crossover and mutation does not allow for finding a valid solution for this case. However, in cases that result in a very small search space, such as `Zero Latency`, it is better to employ an exhaustive search method.

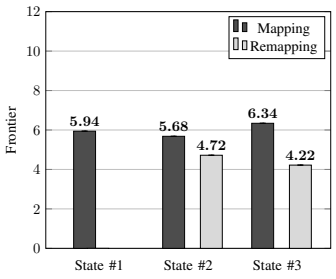

Fig. 8. Variation (T.5)

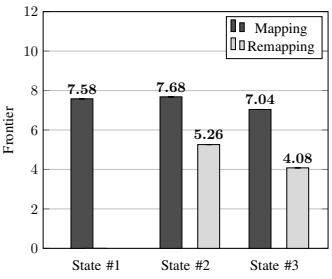

Fig. 9. Variation (T.10)

Finally, the fourth experiment was executed to evaluate the gains related to remapping services when significant gradual changes occur in the network. In this case, the network state was modified at two points on a continuous timeline, resulting in the following states: `State #1`, initial network state where a service is mapped; `State #2`, where 20% of connections (randomly selected) show a decrease in inter-domain latency, triggering a mapping/remapping of the service, and; `State #3`, where 20% of connections (randomly selected) show an increase in inter-domain latency, triggering a new mapping/remapping. Remapping uses the insertion mechanism to insert the last Pareto frontier found as the initial population, while mapping uses the generation mechanism to create new individuals.

Figures 8 (5 functions) and 9 (10 functions) show the results of candidate mappings, in which each bar indicates the average relative Pareto frontier of the individuals returned as a result after 8000 generations. It can be observed that the use of individuals previously optimized for a specific network state as the initial population (remapping) has a positive effect on the quality of the results generated by the solution. In other words, when faced with gradual changes in the network and/or service, historical data can serve as a solid basis for defining starting points for exploring and deepening the search space,

since a solution optimized for a previous state may remain fully or partially viable. However, it is relevant to note that this behavior is not expected in the event of catastrophic changes to the network or service, in which case it would be preferable to discard the historical data and perform a new mapping process.

The experiments presented in this section demonstrate the feasibility of using the proposed solution for mapping network services in environments supported by the Multi-SFC. The scalability of the proposed solution is evidenced by the linear progression of the execution times as the problem grows. Finally, the ability to handle dependencies, especially those related to the Multi-SFC (such as domain type and orchestrator dependencies), along with the ability to perform remapping processes are other highlights of the proposed solution.

V. Related Work

Recent work has focused on the applicability of genetic algorithms in solving the problem of mapping network service chains in distributed environments [21]. Although highly effective, few solutions based on genetic algorithms have been proposed in this context. Two prominent alternatives are GA+LCB [22] and GeSeMa [21], described next.

GeSeMa uses a genetic strategy with a customizable objective function based on SPEA2 to map virtualized services in multi-datacenter environments. GA+LCB employs its particular single-objective genetic strategy to optimize an index created by maximizing link availability, inter-domain bandwidth, domain availability, and minimizing inter-domain delay. However, GeSeMa and GA+LCB applications are limited because of the lack of support for defining domain and orchestrator dependencies, and the lack of allocation and cost calculation of tunneling functions between domains.

In [23], a genetic algorithm was proposed to select and share VNFs to reintegrate service chains in distributed virtualized environments. The algorithm is used to optimize computational resources and the end-to-end latency of the service. However, this solution is subject to the same limitations discussed above. In addition, the genetic heuristic proposed by the authors does not include a mechanism to take advantage of historical data during the execution of service reincorporation, unlike what is done by the SeMENTE solution.

In addition to mapping solutions based on genetic strategies, it is important to highlight those dedicated to the redeployment of virtualized services. NFV-PEAR [24] is a solution based on an ILP modeling of the problem that focuses on the continued recomposition of SFCs. The objective of this strategy is to minimize the amount of resources consumed and modifications of service mappings while maintaining the same performance levels, dependencies, and constraints.

VI. Conclusion

The growing complexity of virtualized network services associated with the constant evolution of NFV solutions makes it inevitable that SFCs will be distributed across multiple domains, clouds, and NFV orchestrators. Thus, it is critical to have a service mapping process that meets cost and performance constraints. This paper presents SeMENTE, a bio-inspired strategy for mapping virtual services in a Multi-SFC scenario. The solution also provides dynamic SFC remapping after network changes. Experimental results show the effectiveness of SeMENTE and demonstrate its potential to improve the process of deploying and managing distributed Multi-SFCs under different scenarios. Future work includes integrating a monitoring solution to SeMENTE so that Multi-SFC remapping can be automatically triggered after pre-specified conditions are met.

References

[1] M. Chiosi *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, 2012.

[2] J. Herrera *et al.*, "Resource allocation in nfv: A comprehensive survey," *Transactions on Network and Service Management*, vol. 13, 2016.

[3] V. Fulber-Garcia *et al.*, "Network service topology: Formalization, taxonomy and the custom specification model," *Computer Networks*, vol. 178, 2020.

[4] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," Internet Requests for Comments, IETF, Tech. Rep., 2015.

[5] M. Ghaznavi *et al.*, "Distributed service function chaining," *Journal on Selected Areas in Communications*, vol. 35, 2017.

[6] OpenStack Foundation, "Tacker: Openstack nfv orchestration," 2024. [Online]. Available: https://wiki.openstack.org/wiki/Tacker

[7] E. T. S. Institute, "Osm: Open source mano," 2024. [Online]. Available: https://osm.etsi.org

[8] G. Venâncio *et al.*, "Beyond vnfm: Filling the gaps of the etsi vnf manager to fully support vnf life cycle operations," *Int. Journal of Network Management*, vol. 31, 2021.

[9] A. Huff *et al.*, "A holistic approach to define service chains using click-on-osv on different nfv platforms," in *Global Communications Conf.*, 2018.

[10] T. Tavares *et al.*, "Niep: Nfv infrastructure emulation platform," in *Int. Conf. on Advanced Information Networking and Applications*, 2018.

[11] A. Huff *et al.*, "Building multi-domain service function chains based on multiple nfv orchestrators," in *Conf. on Network Function Virtualization and Software Defined Networks*, 2020.

[12] V. Fulber-Garcia *et al.*, "Customizable deployment of nfv services," *Journal of Network and Systems Management*, vol. 29, 2021.

[13] M. Luizelli *et al.*, "The actual cost of software switching for nfv chaining," in *Symp. on Integrated Network and Service Management*, 2017.

[14] P. Ngatchou *et al.*, "Pareto multi objective optimization," in *Int. Conf. on Intelligent Systems Application to Power Systems*, 2005.

[15] YAML Organization, "Yaml: Yaml ain't markup language," 2024. [Online]. Available: https://yaml.org

[16] V. Fulber-Garcia *et al.*, "Cusco: a customizable solution for nfv composition," in *Int. Conf. on Advanced Information Networking and Applications*, 2020.

[17] Platypus Organization, "Platypus - multiobjective optimization in python," 2024. [Online]. Available: https://platypus.readthedocs.io

[18] A. Seshadri, "A fast elitist multiobjective genetic algorithm: Nsga-ii," *MATLAB Central*, vol. 182, 2006.

[19] J. Chacón *et al.*, "Analysis and enhancement of simulated binary crossover," in *Congress on Evolutionary Computation*, 2018.

[20] A. Lotov *et al.*, "Visualizing the pareto frontier," in *Multiobjective Optimization: Interactive and Evolutionary Approaches*, 2008.

[21] V. Fulber-Garcia *et al.*, "Customizable mapping of virtualized network services in multi-datacenter environments based on genetic metaheuristics," *Journal of Network and Systems Management*, vol. 31, 2023.

[22] P. Rodis *et al.*, "Intelligent network service embedding using genetic algorithms," in *Symp. on Computers and Communications*, 2021.

[23] Z. Chen *et al.*, "Delay optimization oriented service function chain migration and re-deployment in operator network," *ACTA ELECTONICA SINICA*, vol. 46, 2018.

[24] G. Miotto *et al.*, "Adaptive placement & chaining of virtual network functions with nfv-pear," *Journal of Internet Services and Applications*, vol. 10, 2019.