# A Multi-Objective SFC Placement Scheme over Multiple Domains

Nassima Toumi[1,2], Djamal-Eddine Meddour[1], Adlen Ksentini[2]
[1] Orange Labs Networks, Lannion, France
[2] EURECOM, Sophia-Antipolis, France
Email: {nassima.toumi, djamal.meddour}@orange.com, adlen.ksentini@eurecom.fr

***Abstract*—Service Function Chaining (SFC) has gained momentum in the wake of the emergence of promising technologies such as Network Function Virtualization (NFV), and Software Defined Networking (SDN). Many research works have explored Service Function Chaining under different aspects: composition, resource allocation, as well as placement and chaining. However, SFC mapping in a multi-domain context with limited visibility on the underlying infrastructure is still a challenging research topic. In this work, we model the multi-domain SFC placement problem as a multi-objective ILP, then propose a scalable memetic algorithm. The efficiency of our solution is evaluated by comparing its results to the ones obtained from the exact solution with full visibility on the network topology.**

***Index Terms*—SFC, Multi-domain, NFV, SDN**

## I. Introduction

New generations of networks, including 5G, rely mainly on two network softwarization technologies: Network Function Virtualization (NFV), and Software Defined Networking (SDN) [1]. SDN decouples the control plane from the data plane to enable dynamic network programming; on the other hand, NFV aims at running Network Functions on top of a virtualized environment (i.e. VMs or containers) hosted in the Cloud (central or edge) in order to reduce Capital and Operational Expenditures, and improve service flexibility. Along with NFV emerged the concept of micro-services, which is the decomposition of services in small blocks that perform simple functions, thus creating the need for steering traffic between those functions in a certain order so that services are delivered properly; this process is referred to as Service Function Chaining (SFC), it leverages on SDN and NFV, and brings out many orchestration challenges such as service composition, placement, or traffic steering [2]. More specifically, one challenging research topic in SFC is multi-domain placement and chaining, which occurs in scenarios where a service requires its components to be deployed on multiple domains; this task gets more difficult if the domains belong to different administrative entities that are reluctant to disclose detailed information on their topology; indeed, the lack of sufficient information leads to sub-optimal placement decisions. In this paper, we devise a novel solution for SFC mapping on multiple administrative domains, with a limited view on the network topology, which aims to optimize the cost and end-to-end latency according to the users' Service Level Agreement (SLA). We first model an exact solution using a multi-objective Integer Linear Program (ILP), then introduce a heuristic that ensures scalability.

The contributions of this paper are twofold: *i)* We present a hierarchical placement scheme with limited visibility over multiple domains that supports complex non-linear SFCs, with a backtracking mechanism to handle local placement failure. *ii)* We model multi-domain SFC placement as a multi-objective ILP, then propose a scalable and efficient memetic algorithm that performs SFC mapping according to the client's SLAs.

The remaining of this paper is organized as follows: Section II discusses previous works; Section III presents an overview of the adopted architecture for multi-domain orchestration, as well as the ILP formulation of the problem, and the proposed memetic algorithm; Section IV provides the methodology for our experimental evaluation, and discusses the obtained results; finally, we conclude the paper in Section V.

## II. Related Works

A large set of works have tackled the SFC placement issue, with different optimization parameters such as cost, energy, or latency (refer to [2] for a comprehensive survey); however, most of these works assume that the orchestrator has full visibility and control on the underlying network infrastructure. As previously outlined, deploying service chains on multiple administrative domains adds more constraints to the placement problem. Indeed, for security reasons, the Infrastructure as a Service (IaaS) providers withhold details on their local infrastructure, which makes it difficult to determine the optimal end-to-end placement and chaining of services due to the lack of information. A few works have addressed the multi-domain SFC deployment issue with limited visibility on the network; two main architectural approaches have been proposed:

*a) Distributed:* It supposes that the infrastructure providers don't share any details on their network; in that case, a distributed algorithm is executed on all of the domains, and messages are exchanged in order to determine the best option without disclosing information to external parties. However, this approach falls short in terms of scalability, as communication and convergence time and cost are considerable. The work in [3] details a policy-based, distributed, asynchronous election protocol based on hosting capabilities; the solution allows edge and core cloud providers to cooperatively instantiate wide-area chains; however, the proposed solution does not support more

978-1-5386-8088-9/19/$31.00 

complex, non-linear SFCs, and its placement evaluation only considers CPU and bandwidth constraints. Zhang *et al.* [4] also propose a distributed vertex-centric algorithm that supports SFC flexibility in order, the request is relayed between the orchestrators in order to determine the optimal placement combination.

*b) Centralized:* In this approach, a broker/coordinator collects the information disclosed by different IaaS providers, and reconstitutes an abstract global view of the network, the centralized broker performs an initial placement using this abstract view, then partitions the request and relays the sub-requests to the local domains; however, this approach leads to sub-optimal orchestration decisions due to the lack of sufficient information on the infrastructure state. Figueira *et al.* [5], and Guerzoni *et al.* [6] propose hierarchical architectures for multi-domain SFC orchestration, where a centralized main orchestrator interfaces with lower-level domain orchestrators. Dietrich *et al.* [7] leverage on this architectural approach and detail a solution for SFC mapping across datacenters that are operated by multiple Network Function Providers (NFP). The proposed solution allows NFPs to disclose minimal information about their infrastructure and constructs an abstract view of the network topology. The placement is then performed in two stages: graph partitioning and sub-graph mapping; however, the solution doesn't take latency into account, which is a critical requirement for the upcoming 5G use cases (i.e. Ultra Reliable Low Latency services); furthermore, the solution is formulated as a Linear Program and therefore lacks scalability, which makes it unsuitable for bigger instances of the problem. Similarly, Xu *et al.* [8] propose a multi-domain service chain partition and embedding scheme using a Hidden Markov Model and a Viterbi-based heuristic. The proposed solution only considers latency during the graph partitioning phase, and only aims to minimize cost during the sub-graph mapping phase while discarding latency; which leads to sub-optimal solutions.

## III. Proposed Solution

### A. Architecture

As stated earlier, we adopt a hierarchical approach where a logically centralized multi-domain orchestrator establishes an abstract view of the network using the information disclosed by each domain, and interacts with the different local domain orchestrators as well as the WAN domain operators in order to provide an end-to-end visibility and control of the different SFCs; as in [9], we suppose that the local domains disclose the following information on their infrastructure:

- The total available computing capacity, as well as the average cost per unit for each resource type.
- The vertices of their inter-domain links, their available capacity, latency, and cost per bandwidth unit.

We assume that the amount of computing and link resources made available by each domain/WAN operator is defined by pre-established mutual agreements. Upon receiving a SFC request, the multi-domain orchestrator performs an initial placement of the request using the abstracted view of the network; the placement takes into account the request's link and resource requirements, as well as affinity and anti-affinity constraints between Virtual Network Functions (VNFs), their allowed placement locations, and the level of priority of each optimization objective obtained from the client's SLA. The broker then proceeds to partition the request according to the initial abstract placement, and adds *dummy* boundary nodes at the sub-chain's extremities. These boundary nodes would require to be placed on the border routers of the domain, which serves to direct the traffic out of the domain and to the next sub-chain. The sub-requests are then transferred to the selected domains, and placed by their respective local orchestrators with full knowledge of the topology. In case the placement of a sub-request on a local domain fails, a backtracking mechanism is triggered if possible, in order to re-run the abstract placement operation while ruling out the previous solution, and the graph-partitioning operation is performed again; otherwise, if all of the solutions have been ruled out, the placement request is rejected. After receiving the placement results of the different sub-requests, the end-to-end cost and latency are computed, and a confirmation is sent to the local domain orchestrators in order to trigger SFC deployment. The pseudo code of the abstract placement and request partition scheme is provided in Algorithm 1.

### B. Problem Formulation

We present in this section our ILP formulation of the placement problem. The network infrastructure is modeled as a weighted undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \mathcal{N}_d \cup \mathcal{N}_s$ is the set of vertices representing the forwarding devices (switches and routers) $\mathcal{N}_s$, and the Data Center nodes $\mathcal{N}_d$ with associated computation, memory and storage capacities; the vertices are connected by edges representing physical links from the set $\mathcal{L}$, that are characterized by bandwidth capacity, as well as the induced latency. We will also denote by $\mathcal{P}$ the set of pre-determined physical paths between the physical nodes. We denote by $\mathcal{S}$ the set of SFCs, and by $\mathcal{V}_i$ the set of VNFs in the SFC $i$.

Each SFC request is modeled as a directed weighted graph, VNFs are represented by the vertices with associated resource requirements as well as a set of authorized nodes $\mathcal{M}_{i,j}$ on which the VNFs can be placed, they are connected by the graph's edges, with associated link requirements. We specify bandwidth requirements for each individual link, as some VNFs may change SFC's traffic volumes; the requests also specify the user's preferences regarding the optimization objectives (i.e. cost and latency) according to their SLA. Further, our model supports more complex non-linear SFCs where traffic flows through certain VNFs more than once using *dummy* nodes that are bound to the original VNFs using affinity constraints in order to route the traffic back to these particular VNFs; we assume that the amount of computing resources allocated to each VNF has been adjusted to the volume of processed traffic during the resource allocation phase, which is out of the scope of this work.

**Algorithm 1:** Multi-Domain Placement Algorithm

**Input :** Abstract Topology $\mathcal{G}_{abs}$ , Request $req$
Set of Authorized Domains $\mathcal{M}$
**Output:** Vector containing the placement result

absractPl ← **abstractPlacement**($req$,$\mathcal{G}_{abs}$,$\mathcal{M}$);
**if** *abstract placement failed* **then**
  **return** *False, null, null*
**end**
*// Perform request partitioning*
partReqs ← **requestPartitioning**($req$,abstractPl);
**for** $p \leftarrow 0$ **to** |*partReqs*|*-1* **do**
  partialPl[p] ← **localPlacement**(partReqs[p]);
  **if** *local placement fails* **then**
    *// Perform backtracking*
    **if** $\exists vnf \in \bigcup_0^p partReqs[n], |\mathcal{M}_{vnf}| > 1$ **then**
      Find one vnf for which $|\mathcal{M}_{vnf}| > 1$ ;
      $\mathcal{M}_{vnf} \leftarrow \mathcal{M}_{vnf} - \{$abstractPl[vnf]$\}$;
      Go to Step **1**;
    **else**
      **return** *False, null, null*
    **end**
  **end**
**end**
*// Compute E2E cost & latency*
**for** $i \in$ *['Cost','Latency']* **do**
  $eval[i] = \sum_{p=0}^{|partReqs|-1}$ partialPl[p][i] $+$
  $\sum_{n=1}^{|\mathcal{M}|-1} \mathcal{G}_{abs}$[absPl[n-1]][absPl[n]][$i$];
**end**
Send placement confirmation to local domains;
Update $\mathcal{G}_{abs}$ ;
**return** *True,cost,latency*

*1) Constraints:*

*a) Placement Constraints:*

*Node Mapping:* The boolean variable $\mathcal{X}_{i,j}^{n}$ expresses whether the $j^{\text{th}}$ VNF of the SFC $i$ has been mapped to the physical node $n$.

$$\mathcal{X}_{i,j}^{n} = \begin{cases} 1 & \text{If VNF } j \text{ of SFC } i \text{ is placed on node } n \\ 0 & \text{Otherwise} \end{cases}$$

$$\sum_{n \in \mathcal{M}_{i,j}} \mathcal{X}_{i,j}^{\text{n}} = 1, \qquad i \in \mathcal{S}, \forall j \in \mathcal{V}_i \tag{1}$$

$$\mathcal{X}_{i,j}^{\text{n}} - \mathcal{X}_{i,k}^{\text{n}} = 0, \qquad i \in \mathcal{S}, j,k \in \mathcal{V}, n \in \mathcal{N}_d \tag{2}$$

$$\mathcal{X}_{i,j}^{\text{n}} + \mathcal{X}_{i,k}^{\text{n}} \leq 1, \qquad i \in \mathcal{S}, j,k \in \mathcal{V}, n \in \mathcal{N}_d \tag{3}$$

$i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_j, \forall m \in \mathcal{M}_{j+1}$:

$$\mathcal{X}_{i,j}^{\text{n}} \cdot \mathcal{X}_{i,j+1}^{\text{m}} \leq \rho_{i,n,m}, \tag{4}$$

Constraint 1 ensures that each VNF is mapped to only one physical node, which is an authorized node for this specific VNF. Constraints 2 and 3 express affinity/co-location (i.e. two VNFs must be placed on the same node) and anti-affinity/anti-location requirements. We denote by $\rho_{i,n,m}$ the number of physical paths between the nodes $n$ and $m$ that are allowed for SFC $i$, constraint 4 ensures that two successive VNFs are mapped to two nodes if and only if there is at least one allowed physical path between these nodes. Constraint 4 is quadratic, it can be linearized by introducing a new boolean variable $\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}}$ which is the product of $\mathcal{X}_{i,j}^{\text{n}}$ and $\mathcal{X}_{i,j+1}^{\text{m}}$:

$$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} = \mathcal{X}_{i,j}^{\text{n}} \cdot \mathcal{X}_{i,j+1}^{\text{m}} \tag{5}$$

Constraint 4 is therefore translated to the following:

$$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} \leq \mathcal{X}_{i,j}^{\text{n}} \tag{6}$$

$$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} \leq \mathcal{X}_{i,j+1}^{\text{m}} \tag{7}$$

$$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} \geq \mathcal{X}_{i,j}^{\text{n}} + \mathcal{X}_{i,j+1}^{\text{m}} - 1 \tag{8}$$

$$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} \leq \rho_{i,n,m} \tag{9}$$

*Link Mapping:* We will denote by $\mathcal{P}^{n,m,q}$ the $q^{\text{th}}$ physical path between nodes $n$ and $m$. The boolean variable $\mathcal{Z}_{i,j,j+1}^{n,m,q}$ determines whether the logical link between the $j^{\text{th}}$ VNF and its successor in the SFC $i$ has been mapped to the $q^{\text{th}}$ physical path between nodes $n$ and $m$.

$$\mathcal{Z}_{i,j,j+1}^{n,m,q} = \begin{cases} 1 & \text{If logical link between VNFs } j \text{ and } j+1 \text{ of SFC } i \\ & \text{is mapped to the } q^{\text{th}} \text{ physical path between nodes } n \text{ and } m \\ 0 & \text{Otherwise} \end{cases}$$

$\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_j, \forall m \in \mathcal{M}_{j+1}$ :

$$\sum_{q=1}^{\rho_{n,m}} \mathcal{Z}_{i,j,j+1}^{n,m,q} = \mathcal{X}_{i,j}^{\text{n}} \cdot \mathcal{X}_{i,j+1}^{\text{m}} \tag{10}$$

Constraint 10 ensures that a logical link between two VNFs is mapped to a physical path if and only if the corresponding VNFs are mapped to the nodes that the path interconnects, and vice versa; it also ensures that not more than one physical path is allocated to a logical link. Similarly to 4, the constraint can be linearized as follows:

$$\sum_{q=1}^{\rho_{n,m}} \mathcal{Z}_{i,j,j+1}^{n,m,q} = \dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}} \tag{11}$$

*b) Capacity Constraints:*

*Computing resources:* We will denote by $\Re$ the set of computing resource types of physical nodes (CPU, RAM, disk space...), by $\nabla_{r,i,j}$ the required amount of the resource $r$ for the $j$th VNF of the $i$th SFC, and by $\mathcal{R}_{r,n}$ each node $n$'s remaining capacity for the resource type $r$. Constraint 12 ensures that the total amount of allocated resources on each node for each resource type does not exceed the amount of available resources remaining on the node.

$$\sum_{j \in \mathcal{V}_i} \nabla_{r,i,j} \cdot \mathcal{X}_{i,j}^{\text{n}} \leq \mathcal{R}_{r,n}, \qquad \forall n \in \mathcal{N}_d, \forall r \in \Re \tag{12}$$

*Link resources:* We will denote by $\mathcal{W}_{i,j,j+1}$ the required bandwidth for the virtual link between the $j$th VNF and its successor, and by $\mathcal{R}_{\omega,l}$ each link $l$'s capacity; we will also use the boolean $\tau_l^{n,m,q}$ to express whether the link $l$ is part of the $q$th path between the nodes $n$ and $m$. Constraint 13 ensures that the total allocated bandwidth on each physical link of an end-to-end physical path does not exceed its remaining capacity.
$\forall i \in \mathcal{S}, \forall l \in \mathcal{L}$ :

$$\sum_{j \in \mathcal{V}_i} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \mathcal{W}_{i,j,j+1} \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \leq \mathcal{R}_{\omega,l} \quad (13)$$

*2) Objective Function:* The objective of our model is to minimize the overall cost, as well as the end-to-end latency. However, these objectives are contradictory: physical links' cost is inversely proportional to their latency (i.e. lower latency links are more expensive). A trade-off is then obtained by setting weights to both objectives in order to set optimization priorities according to the user's preferences as specified in the SLAs (uRLLC, best effort, etc.). Therefore, our optimization objective function is a weighted sum of normalized values of the overall cost and the end-to-end latency for each SFC.

$$\mathcal{C}_{computing} = \sum_{n \in \mathcal{N}_d} \sum_{r \in \Re} \sum_{j \in \mathcal{V}_i} \zeta_{r,n} \cdot \nabla_{r,i,j} \cdot \mathcal{X}_{i,j}^{\mathrm{n}} \quad (14)$$

$$\mathcal{C}_{link} = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \zeta_l \cdot \mathcal{W}_{i,j,j+1} \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \quad (15)$$

$$\phi = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \phi_l \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \quad (16)$$

Equations 14 and 15 express computational and link costs for the SFC $i$ respectively, with $\zeta_{r,n}$ being the cost of using the resource $r$ on node $n$ per unit, and $\zeta_l$ the cost of using the link $l$ per bandwidth unit. Equation 16 expresses the end-to-end latency for the SFC $i$, with $\phi_l$ being the link $l$'s latency. The optimization objective is to minimize the following objective function while satisfying the aforementioned constraints.

$$\textbf{minimize} \qquad \alpha_{\mathcal{C}} \cdot \frac{\mathcal{C}_{computing} + \mathcal{C}_{link}}{\lambda_{\mathcal{C}}} + \alpha_{\phi} \cdot \frac{\phi}{\lambda_{\phi}}$$

**subject to**

$$\begin{cases} \text{Constraints 1 - 3} \\ \text{Constraints 6 - 9} \\ \text{Constraints 11 - 13} \end{cases}$$

With $\alpha_{\mathcal{C}}$ and $\alpha_{\phi}$ being the associated weights to cost and latency respectively, which are determined from the users' SLAs; and $\lambda_{\mathcal{C}}$ and $\lambda_{\phi}$ the normalization coefficients for cost and latency, which are determined by computing the mean cost and latency for the topology.

### C. Heuristic Approach

Although the previously detailed ILP model ensures an optimal placement of SFC requests considering the provided information, it lacks scalability; indeed, SFC placement has been proven to be an NP-Hard problem [10], which means that computation time increases exponentially using large problem instances as will be illustrated in our evaluation results in Section IV, thus making this solution impractical for operational use; however, the ILP model's results can be used as a reference in order to evaluate the efficiency of alternative solutions. As an alternative, and in order to support bigger instances of the problem, we propose a memetic heuristic based on a genetic algorithm, combined to a local search method for solution improvement. At first, a set of initial solutions is generated and classified; the set is then updated at each generation by introducing new individuals obtained through solution generation, mutation, crossover, or local search improvement; the set of solutions is then evaluated and classified using the objective function previously expressed in the ILP, and the best individuals are selected for the next generation . The process is repeated until a fixed number of generations is reached, the best solution is then returned; note that all of the operators use intervals in order to reduce request fragmentation. The main steps of our heuristic are:

*a) Solution Generation:* New individuals are generated as follows : the placement of the first VNF is randomly determined from the list of allowed locations, then each subsequent VNF is also placed on the same location as long as the VNFs' affinity, placement constraints, and node's capacity allow it; which reduces link-associated cost and latency. Route mapping is performed by choosing the best available path according to the request's optimization objectives' weights.

*b) Mutation:* During the mutation phase, a solution is randomly selected, then the placement and chaining of a random interval of successive VNFs is changed while respecting the aforementioned constraints.

*c) Crossover:* Two solutions are selected at random, and are crossed at a random interval in order to generate a new solution, VNF chaining is also updated.

*d) Local Search Improvement:* In this phase, a random solution is selected, and its neighbor solutions are explored in order to operate local improvements (e.g. move one or more VNFs to a nearby node).

## IV. Evaluation

### A. Simulation Environment

We evaluated our solution by conducting simulations on the hardware and software configuration outlined in Table I.

| Component | Configuration |
|---|---|
| CPU | Intel Xeon E5-2640 v3 (32 Cores), $2.60GHz$ |
| RAM | $128GB$ |
| OS | Ubuntu 16.04, 64 bits |

TABLE I: Testbed Hardware and Software Configuration

The simulation program and heuristic were developed using the Python programming language; we also used the Gurobi [11] optimizer in order to solve our ILP model.

For the test environment, we generated multi-domain topologies using the *networkx* [12] library, each domain comprises massively connected nodes via intra-domain links, and is able to communicate with the other domains via edge routers connected through WAN links. We also pre-computed the set of paths between nodes for each domain, with a path length limited to 3 hops. After each placement iteration, the graph's resource capacity values are updated. We evaluate our solution using 3 network topologies of different sizes, as detailed in Table II.

| Topology | Small | Medium | Large |
|---|---|---|---|
| Number of Domains | 4 | 6 | 8 |
| Number of Nodes per Domain | 10 | 15 | 30 |
| Number of Links | 166 | 603 | 3388 |

TABLE II: Topology Information

Requests are generated and placed sequentially, each request specifies the amount of required resources and a set of allowed placement locations for each VNF and link, as well as the dependencies between VNFs, and optimization objective's associated weights (cost and latency) obtained from the SLA class of the client. In our simulations, we consider 5 VNF types (small, medium, large, dummy, boundary) with corresponding resource and placement requirements, and 5 SLA classes (ultra low latency, low latency, fair trade-off between cost and latency, low cost, best effort), with different levels of priority for our optimization objectives. The solution is evaluated for SFC lengths ranging from 4 to 10 in the small and medium topologies, and from 4 to 8 in the large topology.

### B. Methodology

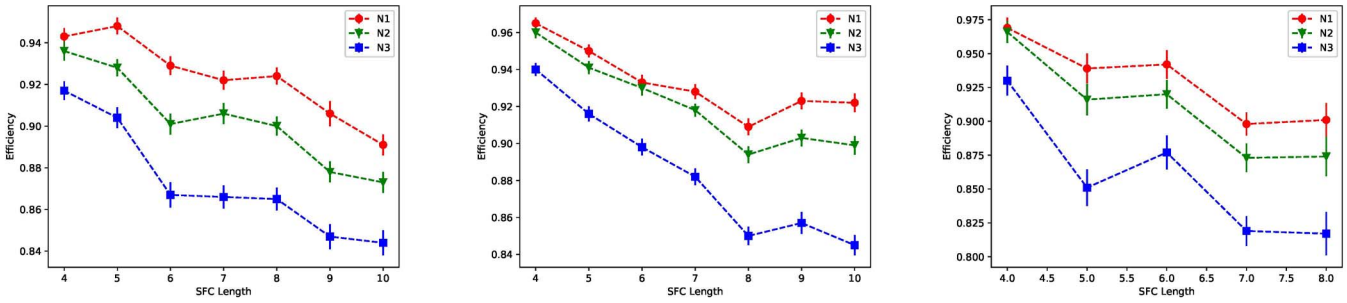
The simulations for each configuration are run 20 times, at each iteration, the placement algorithms are run for 150 successive requests for the small and medium topologies, and 50 requests for the large one. In this evaluation, SFC placement is performed using the ILP with full knowledge of the network topology, as well as the proposed heuristic combined to the multi-domain hierarchical scheme with a limited view on the infrastructure; we vary the number of generations (iterations) of our heuristic in order to evaluate its effect on efficiency and computational time, the number of generations depends on each configuration's parameters, namely the topology size and SFC length : $N_1 = |req| * |\mathcal{G}|$, $N_2 = \frac{N_1}{4}$, $N_3 = \frac{N_1}{16}$. The placement results of the different solutions are evaluated using two key metrics: placement efficiency, which can be computed using the aforementioned objective function; and scalability, which can be quantified by measuring the algorithm's computational time.

### C. Efficiency

Figure 1 illustrates the mean values and $95\%$ Confidence Interval (CI) of the efficiency of our heuristic compared to the exact solution using different topologies and SFC sizes. Efficiency is measured by computing the average ratio between the results of the evaluation of the ILP placement and those obtained from the heuristic placement for each request. We can observe that although the heuristic does not dispose of the full view of the network, nor does it explore the full set of solutions; it displays an accuracy of $94.3 - 89.1\%$ with a $95\%$ CI of $0.8 - 1\%$ in the small network as illustrated in Figure 1a, and $96.5 - 92.2\%$ with a CI of $0.6 - 0.9\%$ in the medium network as shown in Figure 1b when the number of generations is fixed to $N_1$. Whereas in the large topology (Figure 1c), the heuristic's efficiency decreases to $96.9{-}89.8\%$ with a CI of $1 - 3\%$. Increasing SFC length also causes a decrease in efficiency, we can notice a reduction of $4 - 6\%$ when increasing SFC length for numbers of generations $N_1$ and $N_2$ regardless of the topology size; indeed, efficiency merely decreases from $94.3\%$ and $93.6\%$ to $89.1\%$ and $87.3\%$ respectively in the small topology; from $96.5\%$ and $96\%$ to $92.2\%$ and $89.9\%$ respectively in the medium topology, and from $96.9\%$ and $96.6\%$ to $90.1\%$ and $87.4\%$ in the large topology. When setting the number of generations to $N_3$, we notice that increasing SFC length has a more significant impact on efficiency as it is reduced by $7 - 11\%$, reaching a ratio of $84.4\%$ with a CI of $1.2\%$ in the small network, $84.5\%$ with a CI of $1.1\%$ in the medium topology, and $81.7\%$ with a CI of $3.2\%$ in the large one. In contrast, for small SFC lengths, decreasing the number of generations does not have a significant effect on efficiency as it remains above $90\%$ regardless of the topology.

### D. Scalability

Table III features the mean computation times for the ILP and the heuristic solutions when varying topology size and SFC length, as well as the number of generations for the heuristic. The exact solution demonstrates its lack of scalability as it takes up to many hours in order to produce a solution in the large network as well as the medium-sized network when SFC length is increased, which makes it impractical for operational deployment. In contrast, the heuristic proves its scalability as it provides solutions in realistic times regardless of the topology and SFC size: $81ms$ and up to a second in the small topology, $422ms$ to less than 4 seconds in the medium network, and $2.94s$ to $65s$ in the large network when the number of generations is fixed to $N_2$, which is an improvement of up to 600 times compared to the ILP. When setting the number of generations to $N_1$, execution time increases to a range between $241ms$ and $3.46s$ in the small topology, between $1.32s$ and $13s$ in the medium topology, and between $10s$ and $222s$ in the large one. Computational time can be further reduced by setting the number of generations of the heuristic to $N_3$; indeed, it ranges between $37ms$ and $429ms$ in the small topology, $166ms$ and $1440ms$ in the medium topology, and between $994ms$ and $19s$ in the large topology depending on SFC length.




(a) Small Topology (b) Medium Topology (c) Large Topology

Fig. 1: Relative efficiency of the heuristic solution according to the number of generations and topology size

| | | SFC Length | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Topology | Method | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Small | ILP | 1.48 | 2.14 | 3.09 | 5.38 | 7.68 | 11.59 | 17.05 |
| | N1 | 0.241 | 0.350 | 0.631 | 1.03 | 1.25 | 2.22 | 3.46 |
| | N2 | 0.081 | 0.113 | 0.201 | 0.322 | 0.391 | 0.678 | 1.06 |
| | N3 | 0.037 | 0.052 | 0.088 | 0.138 | 0.167 | 0.278 | 0.429 |
| Medium | ILP | 16.29 | 27.74 | 103 | 224 | 376 | 1016 | 2623 |
| | N1 | 1.32 | 1.93 | 3.16 | 4.43 | 7.72 | 10.02 | 13.94 |
| | N2 | 0.422 | 0.588 | 0.938 | 1.30 | 2.21 | 2.88 | 3.98 |
| | N3 | 0.166 | 0.233 | 0.361 | 0.491 | 0.819 | 1.06 | 1.44 |
| Large | ILP | 1381 | 2234 | 3468 | 6434 | 7364 | - | - |
| | N1 | 10.25 | 27.81 | 69.31 | 123 | 147 | 154 | 222 |
| | N2 | 2.94 | 7.92 | 19.47 | 33.55 | 40.57 | 42.13 | 65.47 |
| | N3 | 0.994 | 2.72 | 6.51 | 11.71 | 12.63 | 13.12 | 1976 |

TABLE III: Comparison of computation times (in seconds)

## V. Conclusion

In this paper, we have proposed a multi-objective solution for joint node and link mapping of multi-domain SFCs using an ILP model, as well as a memetic algorithm; the results of our heuristic with limited visibility on the network achieved results that were close to optimal by $96.5 - 89.1\%$ with a computational time improvement of up to 400 times. In future works, we aim to propose a solution that allows a finer grained specification of user preferences, and that provides a Pareto set of the multi-objective solutions.

## References


[1] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "Nfv and sdnkey technology enablers for 5g networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2468–2478, Nov 2017.

[2] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 13, no. 3, pp. 518–532, Sept 2016.

[3] F. Esposito, "Catena: A distributed architecture for robust service function chain instantiation with guarantees," in *2017 IEEE Conf. on Netw. Softwarization (NetSoft)*, July 2017, pp. 1–9.

[4] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Service function chaining in multi-domain networks," in *2016 Opt. Fiber Commun. Conf. (OFC)*, March 2016, pp. 1–3.

[5] N. Figueira and R. R. Krishnan, "Sdn multi-domain orchestration and control: Challenges and innovative future directions," in *2015 Int. Conf. on Comput., Netw. and Commun. (ICNC)*, Feb 2015, pp. 406–412.

[6] R. Guerzoni et al., "Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey," *Trans. on Emerg. Telecomm. Technol.*, vol. 28, no. 4, p. e3103.

[7] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with nestor," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 14, no. 1, pp. 91–105, March 2017.

[8] Q. Xu, D. Gao, T. Li, and H. Zhang, "Low latency security function chain embedding across multiple domains," *IEEE Access*, vol. 6, pp. 14 474–14 484, 2018.

[9] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *2014 IEEE 22nd Int. Symp. of Quality of Serv. (IWQoS)*, May 2014, pp. 61–70.

[10] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov 2017.

[11] Gurobi, http://www.gurobi.com/.

[12] NetworkX, https://networkx.github.io/.