On Using Physical Programming for Multi-Domain SFC Placement with Limited Visibility

Nassima Toumi, *Student Member, IEEE,* Olivier Bernier, Djamal-Eddine Meddour, and Adlen Ksentini, *Senior Member, IEEE,*

Abstract—Service Function Chaining (SFC) is a networking concept by which traffic is steered through a set of ordered functions composing an end-to-end service. It represents one of the facilitating technologies for 5G, and is enabled by the Network Function Virtualization (NFV) and Software Defined Networks (SDN) paradigms. In the multi-domain context, SFC placement faces new challenges related to the lack of visibility on the local domain's networks. Indeed, the domain operators are often reluctant to unveil details on their topology to external parties. Furthermore, the new 5G use cases introduce new requirements for services such as end-to-end latency, and a minimal guaranteed bandwidth that the placement process needs to optimize simultaneously. In this work, we propose a centralized framework that allows SFC partitioning and embedding over multiple domains with a limited visibility over the global infrastructure. We model the multi-objective SFC placement problem using the Physical Programming method, which allows the expression of the Decision Maker's preferences through meaningful parameters, and propose an exact algorithm as well as a scalable heuristic solution. We then perform an extensive evaluation of the framework as well as the proposed algorithms. The results demonstrate our solution's effectiveness with a limited visibility on the network.

Index Terms—Service Function Chaining, Multi-domain, Multi Objective Optimization, Physical Programming, Network Function Virtualization, Software Defined Networks

1 Introduction

The recent advances in Information Technology have introduced a set of new heterogeneous services with specific requirements such as reliability and low latency; furthermore, the advent of massive Machine Type Communication applications has exponentially increased the number of network users, as well as traffic volumes. In this context, a new generation of mobile networks (5G) is under development to meet the requirements of these new use cases [1]; two main enabling technologies have been retained for 5G : Network Function Virtualization, and Software Defined Networking [2]. Leveraging on both of these technologies, Service Function Chaining has emerged as another promising enabler for 5G. It refers to the process of steering traffic between a set of functions in an ordered manner, to deliver an end-to-end service. Service Function Chaining has been studied under different aspects such as orchestration, composition, or placement; however, only a few works have addressed SFC placement in a multi-domain context, which is a challenging research topic since the lack of visibility on the domain's infrastructure hinders the process of obtaining optimal placement results.

On the other hand, optimizing the placement of an SFC in the context of 5G means that different QoS metrics have to be taken into account such as latency and throughput, which leads to the use of Multi-Objective Optimization (MOO) methods, where the most used one is the optimization of weighted sums of each objective [3]. However, this method requires the normalization of the values of each objective, as well as setting weights to assign degrees of importance to each objective. These weights do not precisely express the preferences of users, which means that the obtained solutions won't necessarily reflect the user's preferences.

In this paper, we apply the Physical Programming [4] optimization approach to the multi-objective placement of multi-domain SFCs, which is a method that allows the expression of preferences for different objectives using physically meaningful parameters. We adopt a centralized framework that allows the deployment of SFCs on multiple domains with a limited visibility over their infrastructure. We also propose an exact method for SFC placement, along with a heuristic for scalability purposes. The performance of both solutions is evaluated through extensive simulations. The contributions of this paper are listed below:

- We formulate the multi-domain SFC placement problem using three Physical Programming approaches (Linear, non-linear, global). Three optimization objectives are considered : the end-to-end latency, the bandwidth per user, and the overall cost.
- We propose an exact solution based on a branch and bound algorithm.
- We introduce a scalable heuristic algorithm.
- We evaluate the efficiency of our approach by comparing its results to the ones obtained using a traditional ILP with a weighted sum of the objectives.

N. Toumi, O. Bernier, and D.E. Meddour, are with Orange, 22300, Lannion, France. (E-mail:nassima.toumi@orange.com, ,olivier.bernier@orange.com,djamal.meddour@orange.com)
N. Toumi and A. Ksentini are with the Communication Systems Department, EURECOM, 06410 Sophia-Antipolis, France. (E-mail:nassima.toumi,adlen.ksentini@eurecom.fr)

- We assess the efficiency of our proposed heuristic compared to the exact solution.

To the best of our knowledge, this is the first work that used Physical Programming to formulate the multi-domain SFC placement problem. The remainder of this paper is organized as follows: Section 2 introduces use cases that illustrate the Multi-Domain context of Service Chains. Section 3 summarizes previous related works. Section 4 presents an overview of the proposed framework for multi-domain orchestration. Section 5 features a formulation of the problem, and introduces the Physical Programming method. Section 6 depicts the exact resolution algorithm, while Section 7 describes the proposed heuristic algorithm. Section 8 provides the methodology for our experimental evaluation, and discusses the obtained results. Finally, section 9 concludes the paper.

2 Multi-Domain SFC Use cases

In this section, we present two use-cases of multi-domain SFCs for 5G. The first one is the IoT use case where security, and data aggregation functions are deployed on edge clouds; and the second one is related to the more generic scenario of service composition, where the end-to-end SFC is constructed by deploying and chaining functions on different domains.

2.1 Industrial Internet of Things (IIoT) Use Case

The upcoming 5G networks are set to enable three main usage scenarios: enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLC), and massive machine type communications (mMTC). The latter supports the deployment of a massive number of IoT devices for several applications. Industrial Internet of Things (IIoT) represents one of the most significant applications of IoT, with use cases such as smart factories, that rely on the deployment and the collection of data from a set of IoT devices, e.g. surveillance cameras, sensors etc [5]. Considering the volume of data that is generated by the different devices, the operator could benefit from placing data aggregation functions at the edge of the network, to reduce the amount of consumed network resources [6]. Multiple network providers and vendors have developed commercial solutions that leverage on IoT and edge computing, such as Microsoft with Azure IoT Edge [7], or Amazon with AWS IoT Greengrass [8]. Both solutions allow the deployment of pre-processing functions at the edge of the network, which collect the data generated by the IoT devices and perform filtering and compression before sending it to the core cloud for further processing.

Furthermore, it is essential to protect the infrastructure from different attacks, especially since IoT devices are more vulnerable to hacking and can be used as vectors for large scale attacks [9]. The operator would need to deploy security functions that perform traffic inspection and attack detection, to stop/mitigate attacks such as DDos attacks before they enter the network. Therefore, security functions should also be deployed as close as possible to the sources of data.

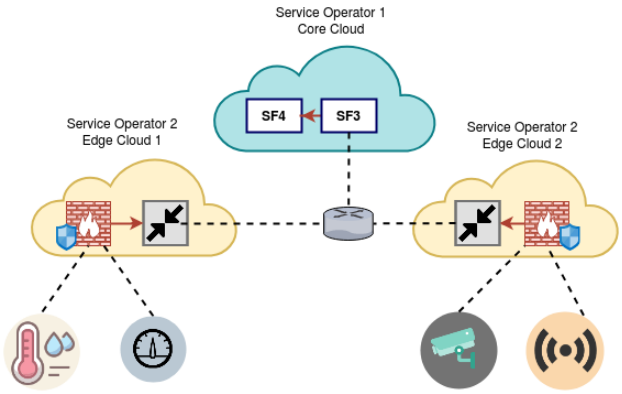


Fig. 1: IoT Use Case: Security and data aggregation VNFs at the edge of the network

However, due to the number and geographical distribution of the devices, the operator may not be able to afford the cost of deploying edge clouds covering the whole sensing area; instead, a less costly alternative is to deploy the security and compression functions on external edge clouds that are closer to the devices. In that case, as illustrated in Figure 1, the operator deploys a multi-domain SFC, where the data flow generated by the IoT devices is directed to the security functions deployed on the external edge clouds. After analysis, the flows that are deemed secure are transmitted to the aggregation functions also deployed on the external edge clouds, before sending the compressed data to the remaining functions of the SFC that are deployed on the operator's domain.

2.2 Service Composition

With the advent of virtualization, different cloud service models emerged : Infrastructure as a Service (Iaas), Platform as a Service (PaaS), and Software as a Service (SaaS); these models enable service providers to grant access to resources, development environment, and software hosted on their infrastructure respectively; thus enabling a multi-tenant multi-layer architecture where physical and virtual resources are shared between different customers (tenants) as independent and isolated slices. In this scenario, as illustrated in Figure 2, the service *tenant* does not own any infrastructure, and composes its own service from a set of Virtual Network Functions (VNFs) that are hosted by different PaaS or SaaS providers. The VNFs are chained together creating the SFC of the tenant, each tenant then disposes of its own overlay slice spanning multiple domains. Note that a slice can host more than one SFC.

3 Related Works

3.1 SFC Placement

A large set of works have tackled the SFC placement issue, with different optimization parameters such as cost, energy, or latency [10] [11]; however, most of these works consider placement over a single domain, and assume that the orchestrator performing the placement disposes of full visibility and control on the underlying network infrastructure. As previously outlined, deploying service chains on

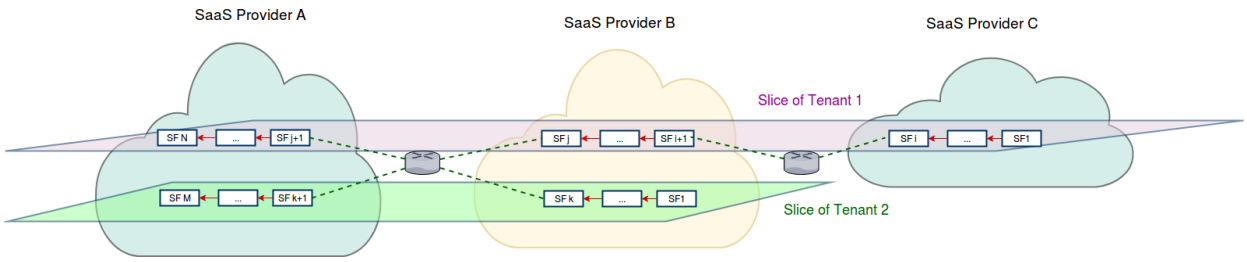


Fig. 2: Service Composition Across Multiple Operator Domains

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 were proposed: distributed, and centralized. Table 1 summarizes the main features of the proposed solutions, and outlines our proposal's contributions. Please note that some works didn't include placement algorithms, which is why the columns related to optimization were left empty.

3.1.1 Distributed

This approach supposes that the infrastructure providers don't share details on their network; in that case, a distributed algorithm is executed on all of the domains, and messages are exchanged 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 important. The work in [12] 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 complex, non-linear SFCs, where packets pass through certain functions more than once; and its evaluation only considers CPU and bandwidth constraints. Zhang *et al.* [13] also propose a distributed vertex-centric algorithm that supports SFC flexibility: the request is relayed between orchestrators to determine the optimal placement combination. The authors in [14] detail DistNSE, a distributed framework that performs SFC partitioning and placement using a privacy-preserving bidding mechanism where each provider competes for the NFs of the chain.

3.1.2 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 might lead to sub-optimal orchestration decisions due to the lack of sufficient information on the infrastructure state. Figueira *et al.* [15], and Guerzoni *et al.* [16] propose a hierarchical architectures for multi-domain SFC orchestration, where a centralized main orchestrator interfaces with lower-level domain orchestrators. Dietrich *et al.* [17] leverage on this architectural approach and detail a solution for SFC mapping across datacenters that are operated by multiple NFPs. 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 some 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.* [18] propose a multi-domain service chain partition and embedding scheme using a Hidden Markov Model and a Viterbi-based heuristic. However, the proposed heuristic solution only considers the end-to-end latency while discarding cost. In [19], the authors propose a multi-domain SFC orchestration scheme that takes into account the issues related to the lack of visibility over domains. They propose an algorithm that aims to minimize the delay, then improves the bandwidth cost of the obtained solution. However, this algorithm does not take into account the service and SLA types of the requests, indeed, a Best Effort service for example would require the minimization of cost over latency.

In this paper, we formulate the multi-domain SFC placement problem, and elaborate multiple algorithms that perform placement with a limited visibility, while minimizing the end-to-end delay and cost, and maximizing the bandwidth per user.

3.2 Multi Objective Optimization

Multi Objective Optimization (MOO) is a process where more than one optimization objective are considered simultaneously. This process requires the articulation of the Decision Maker's (DM) preferences regarding the objectives, which will influence the priority that each objective would have over the others in the optimization process. Many methods have been proposed to express the preferences of the DM before or after the optimization. Most of these methods rely on the assignment of *weights* to each objective, and the optimization of the weighted sum of these objectives [3]. However, these weights are generally determined through trial and error, and do not reflect the specific preferences of the DM as they are not physically meaningful. Furthermore, as each objective has a different scale, a normalization process is required.

Architecture	Solution	Mult. Adm. Domains	Differentiated SLAs	Optimization Objectives			Multi Obj. Opt. Method	Solving Method
				Cost	Latency	Bw		
Distributed	Catena [12]	✓	X	-	-	-	-	ILP
	Zhang et al. [13]	✓	X	-	-	-	-	-
	DistNSE [14]	✓	X	✓	X	X	Weights	Heuristic
Centralized	Figueira et al. [15]	X	X	-	-	-	-	-
	Guerzoni *et al.* [16]	X	✓	-	-	-	-	-
	Nestor [17]	✓	✓	✓	X	X	Weights	ILP/LP
	Xu *et al.* [18]	X	✓	X	✓	X	-	ILP, heuristic
	Sun *et al.* [19]	✓	X	X	✓	✓	-	ILP, heuristic
	Proposed Solution	✓	✓	✓	✓	✓	Physical Programming	Exact algorithm, heuristic

TABLE 1: Multi-Domain SFC Orchestration Solutions and Features

The Physical Programming Optimization approach was proposed by Messac *et al.* in its Non-Linear [4] and Linear [20] forms. It allows the use of physically meaningful parameters to express preferences using ranges for each objective, and different classes that can be hard (H) or soft (S), as shown in Figure 3:

- **Hard Classes:** These classes are constraints by definition, because solutions are rejected if they are not in the acceptable range.
- **Soft Classes:** As illustrated in Figure 4, these soft class functions include 6 preference regions: *Highly desirable*, where improvement past the ideal value is of minimal additional value, *Desirable*, *Tolerable*, *Undesirable*, *Highly Undesirable*, and *Unacceptable*, which is expressed using a constraint.

The range limits provided by the DM, are translated into class functions that allow the evaluation of the obtained solutions in terms of their conformance to the DM's preferences. Multiple methods have been proposed for Physical

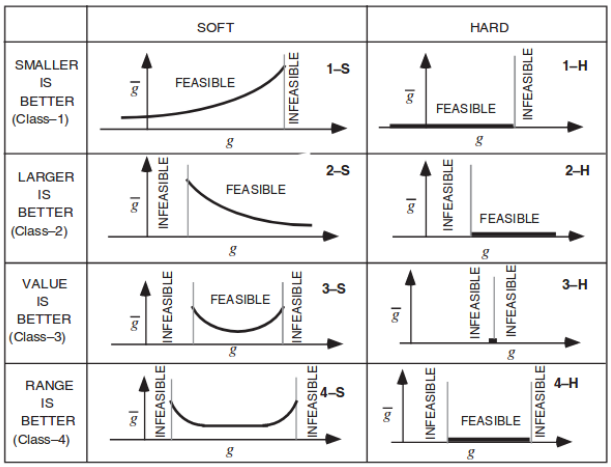


Fig. 3: Physical Programming Preference Classes [4]

Programming: Linear Physical Programming [20] (LPP), Non-Linear Physical Programming (NLPP) [4], and Global Physical Programming (GPP) [21].Each method expresses the class function for each criterion as a combination of piece-wise convex functions using the values of the region boundaries, and the global objective function is defined as the sum of these functions. The main difference between the three methods lies in the complexity of the class function's formulation to obtain a more or less smooth curve, so that

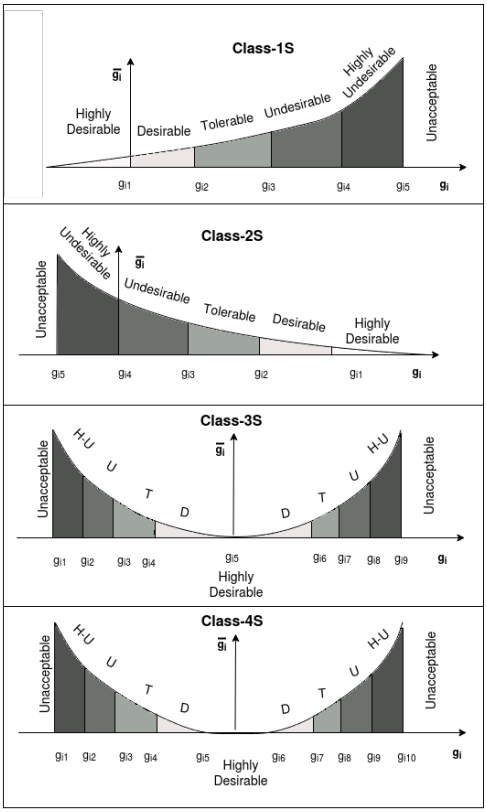


Fig. 4: Region Boundaries For Preference Classes

the class functions translate the preferences of the Decision Maker as accurately as possible. The NLPP formulation allows the most accurate depiction of the Decision Maker's preferences, with a smooth curvature, but at the cost of very complex formulations; the GPP method simplifies the formulation while keeping the Physical Programming properties; and the LPP method allows a linear formulation of the class functions. The approaches for obtaining the objective functions for each Physical Programming method are explained in more details in Section 5.

Physical Programming optimization enforces the One vs Other Criteria (OVO) Rule, which states that a full reduction for one criterion across a given region is better than a full reduction for all of the other criteria across the next better region. In practice, it means that between the options of improving one criterion from the Tolerable region to the Desirable region, and improving all the other criteria from the Desirable to the Highly Desirable region, the first option

would be preferred. This ensures a fair trade-off between objectives, and avoids improving certain objectives at the expense of others, as could be the case with the weighted method. This rule also ensures that the value of the class function for region boundaries is the same across criteria (and therefore that each region's class function's amplitude is also the same across criteria), which has a normalizing effect. Furthermore, once a certain objective has reached the ideal value, instead of further improving it past the ideal value, the focus would be on the remaining objectives.

A few works have leveraged on the Physical Programming method for Multi-Objective Optimization in different fields [22]: aircraft parameter design [23], trajectory planning [24], mechanical engineering [25], as well as electromagnetic and transmission related problems [26]. The results obtained with Physical Programming in each of the aforementioned works are closer to the DM's preferences than the results obtained with the usual MOO methods.

At the time of this writing, and to the best of our knowledge, this approach has not been applied to solve the multi-objective SFC placement problem, further, no existing work compares the different Physical Programming methods. In this paper, we apply three methods of Physical Programming on the SFC placement problem (Linear, Non Linear, Global), and compare their results. The efficiency of Physical Programming is also evaluated by comparing its results to the weighted sum method.

4 Proposed Framework

In this section, we detail our proposed framework for multi-domain SFC deployment. As illustrated in Figure 5, the architecture is logically centralized, with a multi-domain orchestrator that acts as a broker and constructs an abstract view of the topology based on the information that each domain is disposed to reveal on their network. The orchestrator also interacts with the WAN domain operators to establish inter-domain communication for SFCs. Based on the work in [27], we suppose that local domains disclose the following details on their infrastructure :

- The amount of computing capacity that is made available, with the average cost per unit for each resource type. Note that these amounts are determined through pre-established mutual agreements between the domain operators.
- The vertices of their WAN links, the available capacity, latency, as well as the cost per bandwidth unit.

We provide the pseudo-code of the Multi-Domain Placement procedure in Algorithm 1. In this work, we consider that each SFC deployment request is issued by a tenant that will provide the service to multiple users. When the multi-domain orchestrator receives a request req from a tenant to deploy an SFC, it calls the function **abstractPlacement** that performs an initial placement on the previously established abstract view of the topology $\mathcal{G}_{abs}$. This placement takes into account the QoS requirements defined by the SLA of the tenant, as well as the placement constraints of each VNF such as affinity and anti-affinity constraints, and the allowed locations; the optimization objectives and their levels of priority are also defined by the SLA. Once the initial placement $abstractPl$ has been obtained, the SFC request is partitioned accordingly (**requestPartitioning**), and dummy boundary nodes are added at the extremities of the sub-chains. These dummy nodes are required to be placed at the boundaries of the domains, to forward the traffic out of the domains and towards the next sub-chain. Next, the sub-requests $partReqs$ are sent to the chosen domains to perform a local placement **localPlacement**, while having a full view of their respective local topologies; note that each local orchestrator can implement their own placement algorithm. If the placement of a sub-chain fails, a backtracking mechanism is prompted to perform the initial placement once again, but while ruling out the previous placement solution. Once the multi-domain orchestrator receives the placement results of all of the domains, it computes the end-to-end cost and latency of the SFC, then sends a confirmation to the local domain orchestrators to start the sub-SFCs deployment.

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 end-to-end *cost* and *latency*;
Send placement confirmation to local domains;
Update $\mathcal{G}_{abs}$;
return *True,cost,latency*

5 Problem Formulation

We formulate in the following the model of the multi-domain SFC placement problem on both levels. We first define the different placement, resource, latency, and cost constraints. Then, we express the objective function using Linear, Non-Linear, and Global Physical Programming. Table 2 features the notations that are used in our model.

5.1 Constraints

5.1.1 Placement Constraints

A VNF should be placed on only one of its authorized nodes, and only if an available authorized path can be

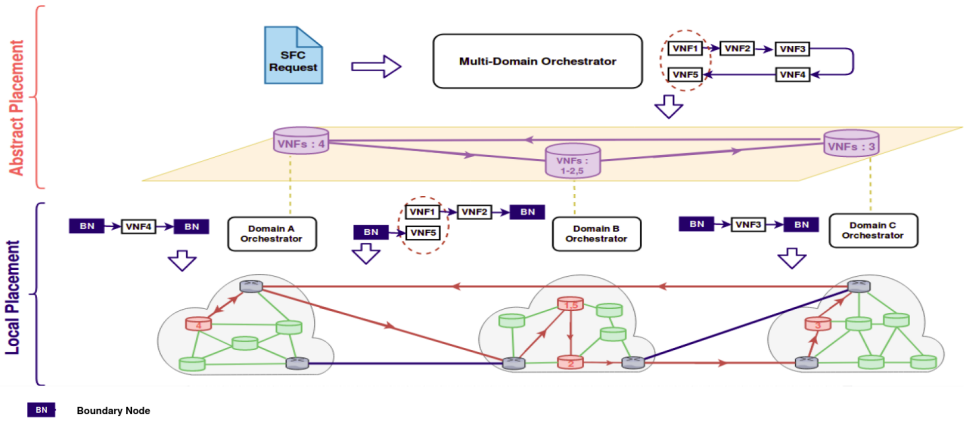


Fig. 5: Multi-domain SFC Placement Framework

Notation	Description
Sets	
$\mathcal{S}$	Set of SFCs
$\mathcal{V}_i$	Set of VNFs in SFC i
$\mathcal{A}_i / \bar{\mathcal{A}}_i$	Set of affinity/anti-affinity constraints for SFC i
$\mathcal{M}_{i,j}$	Set of allowed placement nodes for VNF j in SFC i
$\mathfrak{R}$	Set of computing resources (CPU, RAM, disk)
$\mathcal{N}_d$	Set of nodes in domain d
$\mathcal{L}$	Set of links
Decision Variables	
$\mathcal{X}^n_{i,j}$	Placement of VNF j of SFC i on node n (Boolean)
$\dot{\mathcal{X}}^{n,m}_{i,j,j+1}$	Placement of VNF i of SFC i on node n and its successor on node m (Boolean)
$\mathcal{Z}^{n,m,q}_{i,j,j+1}$	Placement of link between VNFs j and $j+1$ of SFC i on qth physical link between nodes n and m (Boolean)
$\mathcal{W}_i$	Allocated bandwidth per user for SFC i (Integer)
Request	
$\nabla_{r,i,j}$	Required amount of computing resource r for VNF j of SFC i
$\mathcal{U}_i$	Number of users in SFC i
ψ_i^-	Minimal required bandwidth value per user for SFC i
ψ_i^-	Maximal allowed bandwidth value per user for SFC i
C_{paid}	Amount paid by the SFC owner to deploy the Service Chain
Network Infrastructure	
$\rho_{i,n,m}$	Number of physical paths between the nodes n and m that are allowed for SFC i
$\mathcal{R}_{r,n}$	Amount of computing resource r available on node n
$\zeta_{r,n}$	Cost per unit of resource r on node n
$\tau^{n,m,q}_l$	Link l is part of the qth path between nodes n and m (Boolean)
$\mathcal{R}_{\omega,l}$	Capacity of link l
ϕ_l	Latency of link l
ζ_l	Cost of using the link l per bandwidth unit
Physical Programming	
n_{sc}	Number of soft criteria.
g_i	Objective value for criterion i.
$\bar{g}_i$	Class function value for criterion i.
$\bar{s}^k$	Amplitude of interval k on y-axis.
λ_i^k	Amplitude of interval k on x-axis for the criterion i.
g_{is}	Upper/lower limit of interval k on x-axis for the criterion i (Class 1-S/2-S).
β	Convexity parameter

TABLE 2: Notations used

mapped to the other VNFs that it is connected to. Furthermore, additional requirements may mandate placing two VNFs on the same node due to dependencies, or prohibit it, for security concerns. All of these requirements are enforced using the following node and link mapping constraints.

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

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

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

This variable obeys the following constraints:

$$\mathcal{X}^{\text{n}}_{i,k_1} - \mathcal{X}^{\text{n}}_{i,k_2} = 0, \qquad i \in \mathcal{S}, (k_1, k_2) \subset \mathcal{A}_i, n \in \mathcal{N}_d \tag{2}$$

$$\mathcal{X}^{\text{n}}_{i,k_1} + \mathcal{X}^{\text{n}}_{i,k_2} \leq 1, \qquad i \in \mathcal{S}, (k_1, k_2) \subset \bar{\mathcal{A}}_i, n \in \mathcal{N}_d \tag{3}$$

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

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

Constraint 1 ensures that each VNF is mapped to only one of its authorized physical nodes. 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 (i.e. two VNFs must not be placed on the same node), where $\mathcal{A}_i$ and $\bar{\mathcal{A}}_i$ represent the set of tuples of VNFs k_1 and k_2 of SFC i, for which affinity and anti-affinity constraints apply, respectively. 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 consecutive 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 but can be linearized by introducing a new boolean variable $\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1}$ whose value is the product of $\mathcal{X}^{\text{n}}_{i,j}$ and $\mathcal{X}^{\text{m}}_{i,j+1}$, Constraint 4 then becomes:

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

Where the variable $\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1}$ takes the value of the product of $\mathcal{X}^{\text{n}}_{i,j}$ and $\mathcal{X}^{\text{m}}_{i,j+1}$ thanks to the following constraints:

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

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

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

Note that thanks to the affinity constraints, our solution supports non-linear SFC requests. Indeed, if the SFC packet flow is required to pass through a VNF more than once, a new *dummy* VNF instance is created with resource requirements that are null, and the flow is redirected to the original VNF using the affinity constraint that ensures that the original VNF and its dummy instance are placed on the same physical node [28].

Link Mapping: We will denote by q the index of a 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^{th} VNF and its successor in the SFC i has been mapped to the q^{th} physical path between nodes n and m.

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

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

$$\sum_{q=1}^{\rho_{n,m}} \mathcal{Z}_{i,j,j+1}^{n,m,q} = \mathcal{X}_{i,j}^{\text{n}} \cdot \mathcal{X}_{i,j+1}^{\text{m}} \tag{9}$$

Constraint 9 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:

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

5.1.2 Capacity Constraints

SFC placement is also subject to resource capacity constraints. Indeed, a VNF can only be placed on a node that disposes of sufficient computing resources to host it, and a physical path can be used to interconnect two VNFs if and only if enough bandwidth is available on all of its links. These constraints are formulated as follows:

Computing resources: We will denote by $\mathfrak{R}$ 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 jth VNF of the ith SFC, and by $\mathcal{R}_{r,n}$ each node n's remaining capacity for the resource type r. Constraint 11 ensures that the total amount of allocated resources on each node does not exceed the amount of available resources remaining on the node for each resource type.

$$\sum_{j \in \mathcal{V}_i} \nabla_{r,i,j} \cdot \mathcal{X}_{i,j}^{\text{n}} \leq \mathcal{R}_{r,n}, \qquad \forall n \in \mathcal{N}_d, \forall r \in \mathfrak{R} \tag{11}$$

Link resources: We designate by $\mathcal{U}_i$ the number of users of the SFC i, by $\mathcal{W}_i$ the allocated bandwidth per user for the SFC i, 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 qth path between the nodes n and m. We denote by $\mathcal{H}_{i,j,j+1}^{l,q,m,n}$ the amount of bandwidth that is consumed from each link l of the qth path between nodes n and m, by each SFC link between the VNFs j and $j+1$, which is computed as follows: $\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_{i,j}, \forall m \in \mathcal{M}_{i,j+1}, \forall l \in \mathcal{L}$

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} = \mathcal{W}_i \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \cdot \mathcal{U}_i \tag{12}$$

Therefore, $\mathcal{H}_{i,j,j+1}^{l,q,m,n}$ would take one of two values:

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} = \begin{cases} \mathcal{W}_i \cdot \mathcal{U}_i \text{ If the logical link between VNFs } j \text{ and } j+1 \text{ of SFC } i \\ \text{is mapped to the physical path } q \text{ between the nodes} \\ n \text{ and } m, \text{ that the link } l \text{ is part of} \\ 0 \quad \text{Otherwise} \end{cases}$$

Depending on the SLA of the request, $\mathcal{W}_i$ could be a fixed value for the strict SLAs, or variable in cases of SLAs with more relaxed constraints, with a minimal value for the bandwidth. We can therefore discern two different cases :

- **The bandwidth value is fixed:** $\mathcal{W}_i$ is a constant. 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.
  $\forall i \in \mathcal{S}, \forall l \in \mathcal{L}$:

$$\sum_{j \in \mathcal{V}_i} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \mathcal{H}_{i,j,j+1}^{l,q,m,n} \leq \mathcal{R}_{\omega,l} \tag{13}$$

- **The bandwidth value is variable:** In that case $\mathcal{W}_i$ is an integer variable, we will denote by ψ_i^- the minimal amount of bandwidth per user that is required by the SLA of the SFC i, and by ψ_i^+ the maximum amount of bandwidth that is allowed for the SLA of the SFC i. Constraints 14 and 15 ensure that the allocated bandwidth per user for this SFC is within the allowed range of values for the selected SLA of the SFC i:

$$\mathcal{W}_i \leq \psi_i^+ \qquad \forall i \in \mathcal{S} \tag{14}$$

$$\mathcal{W}_i \geq \psi_i^- \qquad \forall i \in \mathcal{S} \tag{15}$$

On the other hand, Constraint 13 becomes quadratic, as $\mathcal{H}_{i,j,j+1}^{l,q,m,n}$ contains the product of a binary variable and an integer variable. It can be linearized using the big M method [29], by replacing Equation 12 by the constraints 16 to 19 to set the value of $\mathcal{H}_{i,j,j+1}^{n,m,q}$ using its upper and lower bounds. $\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_{i,j}, \forall m \in \mathcal{M}_{i,j+1}, \forall l \in \mathcal{L}$:

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} \leq \psi_i^+ \cdot \mathcal{U}_i \cdot \tau_l^{n,m,q} \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \tag{16}$$

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} \leq \mathcal{W}_i \cdot \mathcal{U}_i \cdot \tau_l^{n,m,q} \tag{17}$$

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} \geq \mathcal{U}_i \cdot \tau_l^{n,m,q} \cdot (\mathcal{W}_i - (1 - \mathcal{Z}_{i,j,j+1}^{n,m,q}) \cdot \psi_i^+) \tag{18}$$

$$\mathcal{H}_{i,j,j+1}^{l,q,m,n} \geq 0 \tag{19}$$

5.1.3 Latency Constraint

Depending on the SLA of the SFC owner (Low Latency, Best Effort, etc.), latency constraints may apply. Indeed, a critical service such as autonomous driving would require very low latencies, while a classical web browsing service would allow higher latencies. Equation 20 expresses the end-to-end latency for the SFC i, which is computed as the sum of the latencies of all of the links that are traversed by the SFC packets, with ϕ_l being the link l's latency. Constraint 21 ensures that the end-to-end latency doesn't exceed the maximum allowed value for SFC i, which is denoted by ϕ_i^+.

$$\phi_i = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \phi_l \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \tag{20}$$

$$\phi_i \leq \phi_i^+ \tag{21}$$

5.1.4 Cost Constraint

To ensure a profit margin to the SFC provider, the cost of deploying an SFC should not exceed a certain value. Equations 22 and 23 express computational and link costs for the SFC i respectively. The computational cost $C_{i,computing}$ is calculated as the sum of the costs of allocating the required resources for each VNF of the SFC, depending on the nodes where they have been placed, with $\zeta_{r,n}$ being the cost of

using the resource r on node n per unit. The link cost $C_{i,link}$ is calculated as the sum of the costs of allocating the required bandwidth for each user on each link that has been used to forward the packets of the SFC, with ζ_l being the cost of using the link l per bandwidth unit. The overall cost C_i is the sum of the computational and link cost.

$$C_{i,computing} = \sum_{n \in \mathcal{N}_d} \sum_{r \in \mathfrak{R}} \sum_{j \in \mathcal{V}_i} \zeta_{r,n} \cdot \nabla_{r,i,j} \cdot \mathcal{X}^{\mathrm{n}}_{i,j} \tag{22}$$

$$C_{i,link} = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \zeta_l \cdot \mathcal{H}^{l,q,m,n}_{i,j,j+1} \tag{23}$$

$$C_i = C_{i,link} + C_{i,computing} \tag{24}$$

We will denote by $C_{i,paid}$ the amount paid by the SFC owner in order to have its Service Chain deployed, and by γ the minimal margin of profit that the operator requires. Constraint 25 ensures that the operator keeps its margin above the minimal value.

$$C_i \leq (1 - \gamma) \cdot C_{i,paid} \tag{25}$$

5.2 Objective Function

The objective of our model is to minimize the cost C_i and end-to-end latency ϕ_i, and maximize the allocated bandwidth per user $\mathcal{W}_i$. Since increasing the allocated bandwidth increases cost, and lower latency links are more expensive, the optimization's goal is to find a fair trade-off between all of these objectives while taking into account the tenant's preferences as expressed in the SLAs.

As stated earlier, we use Physical Programming to obtain the multi-criteria objective function. We use the variable i to designate a given criterion, and the variable k to designate a class function interval. Figure 6 features examples of class functions $\bar{g}_i$ of the objective values g_i for all three methods. As illustrated in Figure 6a, we will denote by λ_i^k the x-axis amplitude of each criterion i in a given interval k, and by $\tilde{\bar{g}}^k$ the y-axis amplitude for the interval k. The class function value for each interval's limits is the same across criteria as expressed in relation 26.

$$\bar{g}_k \equiv \bar{g}_i(g_{i,k}) \forall i; (2 \leq k \leq 5); \bar{g}_1 \equiv 0 \tag{26}$$

This means that the y-axis amplitude for each interval $\tilde{\bar{g}}^k$ is also the same across criteria. Note that since the objective of our optimization is to reduce cost and latency, they are expressed using the Physical Programming class 1-S, and bandwidth per user is expressed using the class 2-S as the optimization aims to increase it. Once the function for each criterion i has been defined, the global objective function $G(x)$ to minimize is formulated as follows :

$$G(x) = \log_{10} \left\{ \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} \bar{g}_i[g_i] \right\} \tag{27}$$

The individual class function $\bar{g}_i[g_i]$ for each criterion i is expressed differently depending on the chosen Physical Programming method as will be shown in the following.

5.2.1 Linear Physical Programming

This method has been proposed by Messac *et al.* in [20]. The piece-wise functions for each range are convex and linear as illustrated in Figure 6a. Taking the example of the class 1-S, the piece-wise linear function is expressed as follows :

$$\bar{g}_i = \begin{cases} \bar{g}_{k-1} + \tilde{\bar{g}}^k \left(\frac{g_i - g_{i,k-1}}{\lambda_i^k} \right) & \text{If } g_i \in [g_{i,k-1}, g_{i,k}], 2 \leq k \leq 5 \\ 0 & \text{If } g_i < g_{i,1} \end{cases} \tag{28}$$

The method to compute the values of $\bar{g}^k$ and $\tilde{\bar{g}}^k$ is provided in the Appendix.

5.2.2 Non Linear Physical Programming

This method has been detailed in [4]. The piece-wise functions are convex but non-linear, and can be arbitrarily shaped in order to reflect the priorities of the Decision Maker as illustrated in Figure 6b. This method is the most flexible compared to the other methods, and allows the most accurate depiction of the Decision Maker's preferences, but is also the most complex mathematically. For the sake of readability, we put the mathematical formulations of the class functions in the Appendix.

5.2.3 Global Physical Programming

Global Physical Programming is an adaptation of the Physical Programming method proposed by Sanchis *et al.* [21], which allows a simpler formulation of the problem than the NLPP method, while still satisfying the previously detailed PP rules such as the OVO-rule. It also allows more flexibility in defining N the number of preference ranges for each criterion. The class function for the class 1-S is illustrated in Figure 6c and formulated as follows :

$$\bar{g}_i = \begin{cases} \bar{g}_{k-1} + \tilde{\bar{g}}^k \left(\frac{g_i - g_{i,k-1}}{\lambda_i^k} \right)^n & \text{If } g_i \in [g_{i,k-1}, g_{i,k}], 2 \leq k \leq 5 \\ 0 & \text{If } g_i < g_{i,1} \end{cases} \tag{29}$$

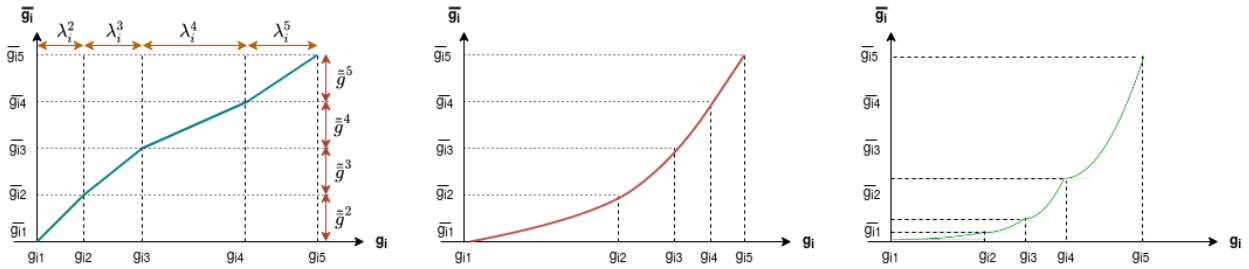
Where n is a pre-defined parameter. Details on how to compute the $\bar{g}^k$ and $\tilde{\bar{g}}^k$ values can be found in the Appendix.

6 Exact Solution

The problem as formulated is a set of piece-wise functions, and contains exponentiations that can reach the value of 4 in the case of Non-Linear Physical Programming. Therefore, the traditional ILP optimizers can't be used to solve the problem. In order to implement the exact resolution of the problem using the given formulation, we propose a Branch and Bound based algorithm that recursively constructs solutions while eliminating branches that don't satisfy the constraints, as well as those with a fitness that is worse than the one of the current best complete solution.

The algorithm takes as input the topology graph $\mathcal{G}$ which comprises the resource information of the nodes and links, the pre-computed set of paths $\mathcal{P}$, the request req which contains the resource requirements for each VNF, the number of users of the SFC, and its SLA class from which we can determine the latency and bandwidth constraints, the set of authorized nodes for placement $\mathcal{M}$ is also provided. The output of the algorithm is the optimal solution $bestSol$.

The solutions are encoded as a vector where the first value represents $\mathcal{W}_i$ the allocated bandwidth per user for



(a) Linear Physical Programming (b) Non Linear Physical Programming (c) Global Physical Programming (n=2)

Fig. 6: Examples of class functions $\bar{g_i}$ of the objective values g_i for the different Physical Programming methods

the SFC i, and each value at the index $2 * j + 1$ represents the ID of the node where VNF j has been placed. Each value at the non-zero index $2 * j$ represents the index q of the path that will be used for the virtual link between the nodes where VNFs $j - 1$ and j have been mapped.

Algorithm 2 features the pseudo-code of the proposed method. First, an initial valid solution is generated, and serves as the best solution reference to eliminate the partial solutions that are already worse. Then, the set of possible solutions is recursively constructed level by level by concatenating values from the set of allowed values per level and evaluating the solution. At the level *zero*, the possible bandwidth values are added, then for each subsequent level, the possible placement nodes per VNF are added, while also computing the best path between the VNF and its predecessor. Once the partial solutions for each level are generated, if they satisfy the constraints, their fitness is calculated according to the SLA of the request and the Physical Programming method that is enforced. This fitness value is then compared to the fitness of the best solution found so far. If the fitness is worse, the solution is withdrawn and the branch is cut, otherwise, if the solution is complete, it becomes the best solution found so far. After all of the possible solutions for a level have been explored, the function *solGen* is recursively called again to compute the solutions of the next level. Once the last level is reached, the best solution is returned.

7 Heuristic Solution

Although the previously detailed 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 [30], which means that computation time increases exponentially using large problem instances as will be illustrated in our evaluation results in Section 8, thus making this solution impractical for operational use; however, the exact algorithm's results can be used as a reference to evaluate the efficiency of alternative solutions. As an alternative, and 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 section 5, and the best individuals are selected for the next generation. The process is repeated until a fixed number of generations is reached, and the best solution is then returned. Note that all of the operations use intervals to reduce request fragmentation. As shown in Figure 7, the main steps of the heuristic are:

- **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 all of the constraints allow it; to reduce link-associated cost and latency. Route mapping is performed by choosing the best available path according to the request's SLA.
- **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.
- **Crossover:** Two solutions are selected at random, and are crossed at a random interval to generate a new solution, VNF chaining is also updated.
- **Local Search Improvement:** In this phase, a random solution is selected, and its neighbor solutions are explored to operate local improvements (e.g. move one or more VNFs to a nearby node).

Similar to the exact algorithm, the function that computes the fitness value of each solution depends on the SLA of the request and the Physical Programming method.

8 Evaluation

8.1 Test Environment

Our simulations were conducted on a server with 32 Intel Xeon 2.60 GHz CPU cores, 128 GB of memory, hosting an Ubuntu Server 16.04 x64 OS. The simulation program, as well as the Branch and Bound and memetic algorithms were developed using the Python language, and the Gurobi solver [31] was used to implement the ILP model.

8.2 Topology and Requests

In order to evaluate our solutions, we leveraged on the *networkx* [32] library to generate multi-domain topologies where each domain is a 3-level Fat Tree network, which is a network topology that is used in data-centers. The physical

Algorithm 2: Branch and Bound Placement

Input : Topology $\mathcal{G}$, Paths $\mathcal{P}$, Request req
Set of Authorized Nodes $\mathcal{M}$
Output: Placement pl

Function $solGen(partS,lvl,\mathcal{M},req,\mathcal{G},\mathcal{P},partC,partL)$ **:**
  $newPartS \leftarrow [] ;$
  for $sol \leftarrow 0$ **to** $|partS|$-1 **do**
    for $j \in \mathcal{M}[lvl]$ **do**
      $currSol \leftarrow [i \text{ for } i \in partS[sol]];$
      if lvl= 0 **then**
        $currSol \leftarrow currSol + [j];$
      else
        Compute $path$ the index of the best path between the last node in $currSol$ and j;
        $currSol \leftarrow currSol + [path, j];$
      end
      if $currSol$ *satisfies the constraints* **then**
        Compute $currFit$ the fitness of solution $currSol$;
        if $currFit < bestFit$ **then**
          $newPartS \leftarrow nwPartS + [currSol] ;$
          if $lvl = maxLvl$ **then**
            $currBestSol \leftarrow currSol;$
            $bestFit \leftarrow currFit;$
          end
        end
      end
    end
  end
  if $lvl = maxLvl$ **then**
    return $currBestSol$
  else
    solGen($newPartS$,lvl+1,$\mathcal{M}$,req,$\mathcal{G}$,$\mathcal{P}$)
  end
end
Generate one valid initial solution $currBestSol$, and save its fitness in $currFit$;
Determine the set of allowed bandwidth values $\mathcal{B}$ from the SLA of the request;
$partSol \leftarrow [[b] \text{ for } b \in \mathcal{B}];$
$lvl \leftarrow 0;$
$bestSol \leftarrow solGen$(partS,lvl,M,req,G,P)
return $bestSol$

servers are connected to edge switches that are connected to aggregate switches, and are in turn connected to core switches that ensure the inter-domain communication. We also use this library to compute the set of paths between nodes. The generated multi-domain topologies are depicted in Table 3. For the link characteristics, we follow the fat tree concept by allocating more bandwidth to the links as we go higher in the topology. In terms of latency, two types of links are used : low latency links, and less expensive high latency links. The servers that host VNFs can be of 3 types with different computing resource capacities, the hardware configurations of each server and link type are detailed in Table 3. Note that these types for each component are randomly selected with equal probabilities.

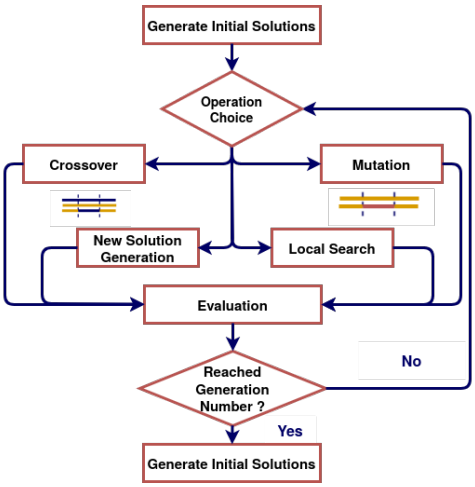


Fig. 7: Proposed Heuristic Solution

Topology Configurations			
Topology	**Domain Number**	**Node Number**	**Link Number**
T1	4	144	240
T2	6	216	368
T3	8	288	496
Link Configurations			
Link Type	**Bw (Gbps)**	**Latency (ms)**	**Price per unit**
Server-Edge	10	0.1-0.5	10-20
Edge-Aggregation	10	0.5-1	40-80
Aggregation-Core (Low Latency)	40	0.5-1	480-960
Aggregation-Core (High Latency)	40	10-20	160-320
Inter-domain (Low Latency)	100	1-5	1920-3840
Inter-domain (Medium Latency)	100	10-15	1280-2560
Inter-domain (High Latency)	100	25-50	640-1280
Node Configurations			
Node Type	**CPU**	**RAM (Gb)**	**Disk (To)**
Large	500	1024	256
Medium	250	256	64
Small	100	64	16
Cost Per unit	10-20	10-20	1-2
VNF Resource Requirements			
VNF Type	**CPU**	**RAM (Gb)**	**Disk (Go)**
Large	4	6	80
Medium	2	4	40
Small	1	2	20

TABLE 3: Topology and Request Characteristics

As for the SFC requests, they are generated with VNF lengths of 4 to 10. Where each VNF can be of one of three types (Large, Medium, Small) with different resource requirements as detailed in Table 3. Each SFC serves as a slice that accommodates a certain number of users, and we randomly set the number of users for each SFC in the interval *10-5000*. In order to support the 5G use cases, we also set different SLA types to each request based on the 5G service types (URLLC, mMTC, eMBB), and different Standard Developing Organizations (SDO) documents [33] [34] [35] [36]. We can identify many SLA classes with different requirements regarding the QoS metrics :

- **Low Latency - High Throughput :** For Augmented or Virtual Reality applications, or live video streaming with strict latency and bandwidth requirements.
- **Low Latency - Low Throughput :** For mission-critical signaling in scenarios such as Autonomous

SLA Class	Use Case	Latency	Bandwidth
0	Real-time video applications (AR,VR,live streaming)	Strict : < 10ms	Guaranteed : > 1 Gbps
1	Mission-Critical Signaling (V2X, Smart Factories)	Strict : < 10ms	Guaranteed : > 10 Mbps
2	Internet Usage, Platinum	< 100ms	> 100 Mbps
3	Internet Usage, Gold	< 200ms	> 50 Mbps
4	Internet Usage, Silver	< 300ms	> 25 Mbps
5	Internet Usage, Best Effort	Best Effort	Best Effort

TABLE 4: SLA Classes

Driving (V2X messages), and Smart Factories, with strict latency requirements.

- **Medium Latency - High Throughput :** For buffered video streaming and file sharing applications.
- **Medium Latency - Low Throughput :** For other TCP-based services such as web browsing, messaging etc.

Additionally, in the general case of classic internet usage, operators provide different offers of level of service to their customers that vary in cost (Platinum, Gold, Silver, Best Effort). In this vein, we set 6 different SLA classes with strict and non-strict QoS requirements as described in Table 4. These requirements will next be translated to Physical Programming classes for use in the multi-objective optimization scheme:

- In the case of SLA classes 0 and 1, latency and bandwidth requirements are strict and are therefore equivalent to Physical Programming hard classes 1-H (Must be smaller) and 2-H (Must be greater) respectively, and they can be modeled as linear constraints.
- For SLA classes 2 to 4, latency and bandwidth requirements are less strict, they can be translated to Physical Programming soft classes 1-S (Smaller is better) and 2-S (Greater is better) respectively.
- SLA Class 5 is best effort, hence only cost is minimized.
- The cost objective for all of the classes will need to be minimized and is equivalent to the Physical Programming soft class 1-S (Smaller is better).

Given these values, and considering that a tenant with a certain SLA shouldn't be able to benefit from a service level that corresponds to a higher nor a lower level than its own, we can define ranges of allowed values for the latency and bandwidth. The deployment cost however always has to be inferior than 95% of the amount paid by the SFC owner irrespective of the SLA class, to guarantee a profit margin of at least 5% to the operator. Once the ranges have been defined, we can define the values of preference intervals for each objective and each SLA class. Table 5 details the intervals for the Physical Programming classes that will be used to generate the evaluation functions of each objective. We denote by - the *Unacceptable* SLA class, by H-U the *Highly Undesirable* SLA class, by U the *Undesirable* class, by T the *Tolerable* class and by D and H-D the *Desirable* and *Highly Desirable* classes respectively.

Additionally, convexity parameters also need to be defined for each Physical Programming method. For NLPP, we set the constant α at 0.05, and the convexity parameter β at 1.5. In LPP, β is set at 1.1, and in GPP, α_{init} is set at 0.1 and the exponent n is set at 2. By applying these values, we

SLA Class	-	H-U	U	T	D	H-D
Latency (ms)						
2	> 105	95-105	90-95	85-90	75-85	< 75
3	> 210	190-210	180-190 ms	170-180	170-150	< 150
4	> 315	285-315	270-285	255-270	225-255	< 225
Bandwidth per User (Mbps)						
2	< 95	95-105	105-115	115-125	125-135	> 135
3	< 47.5	47.5-52.5	52.5-57.5	57.5-62.5	62.5-67.5	> 67.5
4	< 23.75	23.75-26.25	26.25-28.75	28.75-31.25	31.25-33.75	> 33.75
Relative Deployment Cost (%)						
All SLAs	>95	85-95	70-85	60-70	50-60	<50

TABLE 5: Physical Programming Classes

obtain the class functions for each objective and each SLA. Figure 8 illustrates the obtained functions for SLA class 2 for latency, bandwidth, and cost respectively. For each figure, the blue line represent the Linear Physical Programming function, the red line represents the Non-Linear Physical Programming function, and the green line represents the Global Physical Programming function. It can be noticed that the class function results for each objective at the interval limits are equal, which normalizes the objectives.

8.3 Algorithms

In our simulations, we evaluate 3 different optimization algorithms :

- The heuristic combined to the SFC hierarchical partitioning framework and applied to both levels of placement with the Physical Programming methods, and with a limited visibility on the domains at the first step (since the algorithm performs the initial placement on an abstracted view of the topology). Note that the number of generations N for which the heuristic is run is varied, to study its effect on the efficiency of the algorithm, depending on the SFC length and topology size. It would have as values : $N_1 = |req| * |\mathcal{G}|$, $N_2 = \frac{N_1}{4}$, $N_3 = \frac{N_1}{16}$. Where $|req|$ is the length of the request, and $|\mathcal{G}|$ is the size of the topology. Since each local domain only receives part of the original SFC, the cost and latency limits for each sub-SFC are recalculated according to the sub-SFC length's proportion to the original SFC's length.
- In order to evaluate the efficiency of the heuristic algorithm we also implement the Branch and Bound algorithm applied on the full topology with the three Physical Programming methods.
- To evaluate the efficiency of Physical Programming compared to weighted sum MOO methods, we implement an ILP applied on the full topology using constraints expressed in Section 5, but where the objective function is a weighted sum of the normalized objectives:

$$\textbf{minimize} \qquad \alpha_C \cdot \frac{C_s}{\lambda_C} + \alpha_\phi \cdot \frac{\phi}{\lambda_\phi} - \alpha_\mathcal{W} \cdot \frac{\mathcal{W}_i}{\lambda_\mathcal{W}}$$

With α_C, α_ϕ, and $\alpha_\mathcal{W}$ being the associated weights to cost, latency, and bandwidth respectively, which are determined from the tenant's SLA as shown in Table 6. For SLA classes 0 and 1, the latency and bandwidth requirements are strict, which means that they are expressed as constraints, their associated weights are therefore set to 0, and only cost is minimized in the objective function. The SLA classes 2 to 4 represent

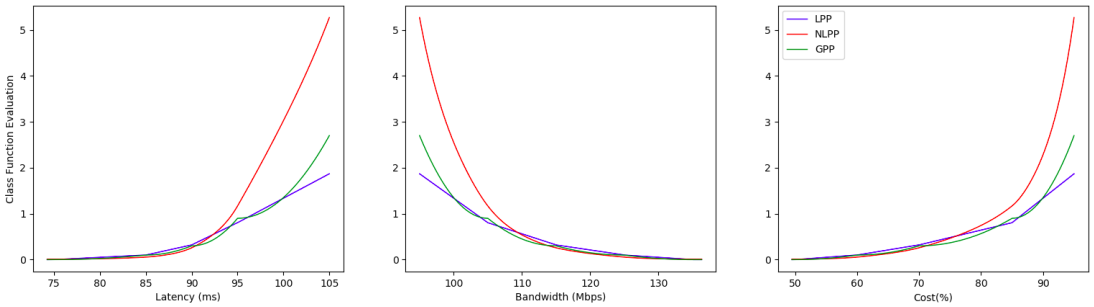


Fig. 8: Physical Programming class functions for the SLA class 2

different levels of service for internet usage, where depending on the subscription, the operator would give more or less priority to the QoS related objectives (allocated bandwidth and latency) over the cost objective. The last SLA class is best effort, which means that only cost is considered, and the weights associated to latency and bandwidth are set to 0. We also denote by λ_C, λ_ϕ, and λ_W the normalization coefficients for cost, latency, and bandwidth which are set as the optimal values for each objective for the selected SLA. The ILP is also applied on the complete multi-domain topology.

SLA	Cost α_C	Latency α_ϕ	Bandwidth α_W
0-1 (Strict QoS reqs.)	1	0	0
2 (Platinum)	0.2	0.4	0.4
3 (Gold)	0.5	0.25	0.25
4 (Silver)	0.8	0.1	0.1
5 (Best Effort)	1	0	0

TABLE 6: Weights associated to the optimization objectives for the ILP depending on the SLA

For the three algorithms, 150 requests of different SLAs are generated and placed sequentially, and after each SFC placement, the capacity values of the topology are updated. Note that since the output of the heuristic can vary from an experiment to another, we repeat the experiment 20 times.

8.4 Results

In the following, we present the results obtained from the different simulations. We evaluate the performance of our algorithms using two metrics: First, we assess efficiency, which can be quantified by classifying each solution in the corresponding SLA satisfaction class according to the obtained results, and comparing the overall results. The second evaluation metric is scalability, which can be estimated by measuring the evolution of runtime for each algorithm when increasing SFC length and topology size.

8.4.1 Efficiency

Overall Satisfaction Rate: First, we evaluate the efficiency of the Physical Programming method compared to the ILP by classifying the placed solutions according to the preference ranges provided by the SFC owner. Table 7 provides the classifications of all of the requests regardless of SFC length and topology type.

It can be observed that while the ILP provides the best placement solutions in terms of latency with 100% of the solutions in the *Highly Desirable* range, it falls short in terms of allocated bandwidth per user where the minimal value is allocated in 100% of the time. For the relative deployment cost, 30% of the solutions are classified at the *Highly Undesirable* range and only 65% of the request placements are classified at the *Highly Desirable* range, even when higher weights are assigned to the cost and bandwidth objectives. In comparison, the Physical Programming results maintain relatively good results across all objectives. Indeed, the Branch and Bound algorithm combined with Physical Programming provides solutions where latency is in the *Highly Desirable* range 91 to 94% of the times, where cost is always kept at a minimal value, and where the optimal value for bandwidth is reached for 95% to 97% of the requests.

It can also be noticed that among the Physical Programming methods, the Linear one is the most efficient. Although all of the methods perform similarly in terms of cost, the LPP method keeps the number of solutions in the *Highly Undesirable* range to a minimum with 0.58% of solutions for latency and 1.08% solutions for bandwidth, as opposed to 1.47% and 4.07% respectively for NLPP, and 1.59% and 2.8% of solutions for GPP respectively. The number of *Highly Desirable* solutions for LPP is also the highest with 94.99% solutions as opposed to 91.31% and 91.62% solutions for NLPP and GPP respectively regarding latency, and 97.15% solutions as opposed to 95.69% and 95.81% solutions for NLPP and GPP respectively in terms of bandwidth. However, this difference remains in the range of 2 – 3% of the produced solutions, which isn't a significant difference.

This difference between the results of the Physical Programming methods and the ILP is due to the fact that in Physical Programming, if a solution reaches the optimal value for an objective, its class function keeps outputting the same value even if the objective is further optimized, thus giving preference to solutions that also improve the other objectives. The weighted-sum ILP however, might give preference to solutions that minimize a certain objective past its ideal value, while disregarding the other objectives.

Since the LPP method performs better than the other two, and for the sake of brevity, we will only display the results of the heuristic using LPP, while varying the generation number. It can be noticed that the solutions from the heuristic are less optimal than those obtained using the Branch and Bound algorithm, with more solutions that are classified on the least desirable ranges; however, the heuristic still outperforms the ILP with results that remain at the *Highly*

SLA Class	H-U	U	T	D	H-D
Latency					
B&B - LPP	0.58%	0.96%	2.29%	1.21%	94.99%
B&B - NLPP	1.47%	2.10%	2.54%	2.61%	91.31%
B&B - GPP	1.59%	2.04%	2.61%	2.16%	91.62%
ILP	0%	0%	0%	0%	100%
H - N1	1.15%	0.77%	0.85%	0.58%	96.68%
H - N2	1.16%	0.88%	0.98%	0.55%	96.46%
H - N3	1.05%	0.78%	0.85%	0.46%	96.89%
Bandwidth per User					
B&B - LPP	1.08%	0%	1.78%	0%	97.15%
B&B - NLPP	4.07%	0%	0.26%	0%	95.69%
B&B - GPP	2.8%	0%	1.40%	0%	95.81%
ILP	100%	0%	0%	0%	0%
H - N1	0.70%	0.38%	0.43%	0.52%	98%
H - N2	0.67%	0.46%	0.61%	0.52%	97.77%
H - N3	0.70%	0.41%	0.60%	0.58%	97.74%
Relative Deployment Cost					
B&B - LPP	0%	0%	0%	0%	100%
B&B - NLPP	0%	0%	0%	0%	100%
B&B - GPP	0%	0%	0%	0%	100%
ILP	29.97%	2.07%	1.57%	1.76%	64.64%
H - N1	0.30%	0.42%	0.29%	0.41%	98.61%
H - N2	0.36%	0.43%	0.36%	0.47%	98.40%
H - N3	0.23%	0.41%	0.31%	0.44%	98.63%

TABLE 7: Results Classification using Preference Classes

Desirable range for 93 – 96% of the placement results. The change in the number of generations does not have a significant effect on the overall results, as the results classification remains consistent with a slight decrease in the number of solutions that are classified in the *Highly Desirable* range, as the number of generations decreases. Similarly to the Branch and Bound algorithm, all objectives are uniformly optimized and no objective is given preference over the two others.

Effect of the SFC Length and Topology Size: Next, we study how the efficiency of the 3 solutions is affected by the changes of SFC length and topology size. Figures 9, 10 and 11 illustrate the obtained results for latency, bandwidth, and cost respectively. The 0 value on the *y-axis* represents the optimal value for each request, and the different results represent the relative mean difference between the obtained value for each algorithm and the optimal one. Depending on the optimization objective, it would be preferable to have results either above or below the optimal value threshold. Indeed, since the optimization model aims to reduce the end-to-end latency and relative cost, the best solutions should be below the threshold. In contrast, as the bandwidth per user is a Larger-is-Better Physical Programming class, the best solutions are above the threshold. The blue line represents the average ILP results, the orange, green and red lines represent the mean Branch and Bound results for the LPP, NLPP, and GPP methods respectively, and the purple, brown and pink dashed lines represent the mean and 95% Confidence Interval (C.I) of heuristic results using LPP for the generation numbers N1, N2, and N3 respectively.

Looking at the end-to-end latency results in Figure 9, we can first confirm the results in Table 7, where the ILP gives preference to latency over the other objectives. The latency values for the ILP are under the optimal threshold with a mean value of around 95−100% less than the objective value across SFC lengths and topology sizes. The same behavior can be observed from the values obtained from the Physical Programming algorithms regardless of the topology size and SFC length. As for the Branch and Bound and heuristic results, we can observe that latency values slowly increase proportionally with SFC length and topology size while still remaining under the optimal value threshold, and therefore within the *Highly Desirable* preference class with values increasing from −81.5% to −38.4% for the topology T1, −80.72% to −29.04% for the topology T2, and −68.75% to −20.33% for the topology T3. It can also be noticed that although all of the Physical Programming plots are similar, the NLPP latency results are slightly lower than those of the other two methods for the Branch and Bound algorithm for the shorter SFCs with values of −83.63%,−80.72%, and −68.75% for SFC length of 4 SFCs and topologies T1,T2, and T3 respectively. However, as SFC length increases, the LPP method provides solutions with slightly lower latencies with values of up to −45.45%,−35%, and −32.81% for SFC length 10 and topologies T1,T2, and T3 respectively. Another observation that can be made on the heuristic results is that the highest generation numbers N1 and N2, provide, on average, lower latencies than those obtained with the generation number N3. This can be explained by the fact that the longer a heuristic is run, the more solutions are explored and the better results can be. In contrast, the ILP results for bandwidth (Figure 10) remain stable and under the optimal value threshold for all SFC lengths and all topologies with mean values of around −25%. Note that in Table 7 all of the placement solutions are classified as *Highly Undesirable*. As for the Branch and Bound results, they also remain stable but slightly over the optimal value threshold, which is consistent with the results in Table 7 where the majority of solutions are classified as *Highly Desirable*. The NLPP method provides the highest bandwidth values with an average of values that are 8% higher than the optimal value for lower SFC lengths, and slowly decreasing to 0.9% as the number of VNFs increases, as opposed to 6 – 2% for LPP and 7 – 0.1% for GPP.

However, the bandwidth values for the heuristic algorithm fluctuate more depending on SFC length and topology size. Indeed, for topology T1, the mean values of bandwidth per user for the heuristic algorithm are higher than the optimal value, and those of the Branch and Bound solution; the values oscillate between 0.04% and 79% over the *Highly Desirable* threshold, with C.Is of 0.4−21% respectively, with a general tendency of decreasing as the SFC length increases. The heuristic results for bandwidth stabilize as topology size increases. For topology T2, the mean values of the bandwidth per user are reduced, but are still close to the optimal value with results that are between 6% below the threshold, and 23% over the threshold, and C.Is of 0.4−12%. As for topology T3, the mean bandwidth values are very similar to those of the Branch and Bound algorithm for the shorter SFCs with values of around 2 – 4%, and a peak at SFC lengths 9 and 10 that reaches 40% over the optimal value with a C.I of 11% for generation number N3; while the mean values for generation numbers N1 and N2 reach the values 20% and 24% over the threshold respectively for the SFC length of 10 VNFs, with C.Is of 8% and 9% respectively. Moving on to the relative cost, we can see that once again the Branch and Bound results remain stable and way under the optimal value across configurations with

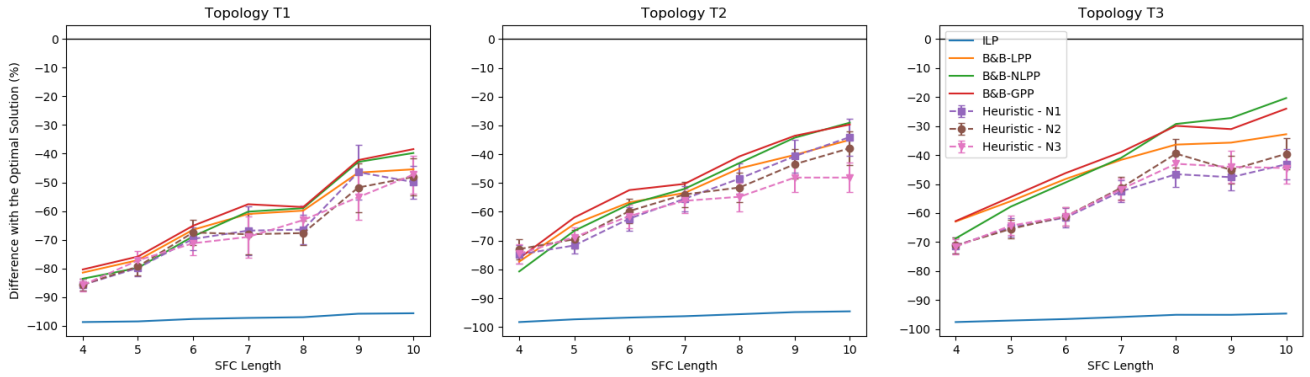


Fig. 9: Relative Difference Between the Placement Solutions and the Optimal Value for the End-to-End Latency

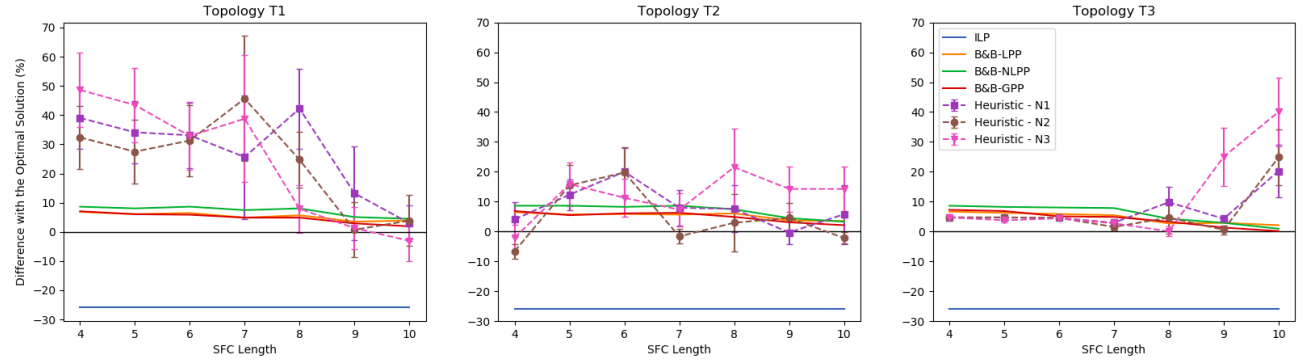


Fig. 10: Relative Difference Between the Placement Solutions and the Optimal Value for the Bandwidth per User

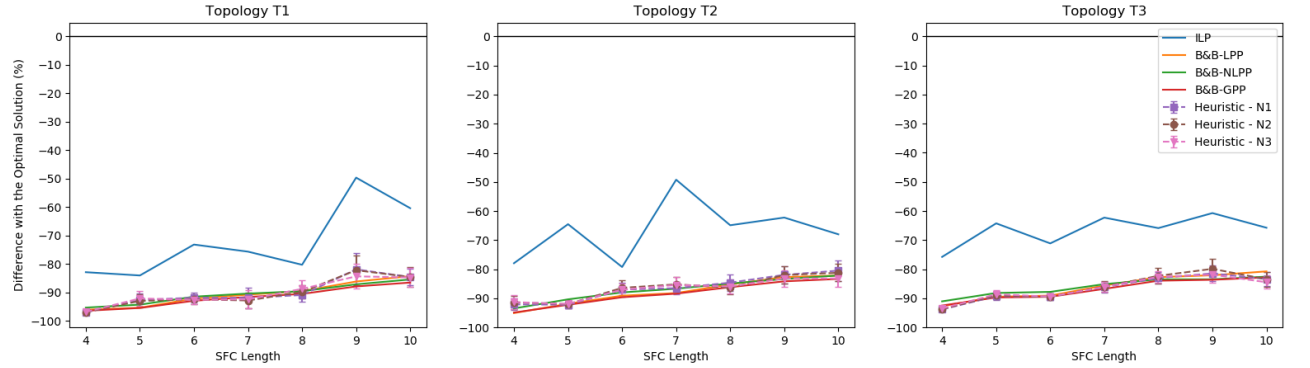


Fig. 11: Relative Difference Between the Placement Solutions and the Optimal Value for the Relative Cost

mean values of −96% to −80%. We can also observe that for this objective, the GPP method slightly outperforms the LPP and NLPP methods. The heuristic algorithm also displays values that slowly increase as SFC length increases, but the results remain similar to the ones obtained by the Branch and Bound algorithm. In contrast, as the ILP provides lower latency values, hence using higher-cost links compared to the other solutions, its cost results are higher than those of the algorithms that leverage on Physical Programming. For topology T1 the relative cost increases proportionally to the SFC length with a mean relative cost of −82% to −49%; while for topology T2 the ILP provides results with relative costs of 79% to 49% lower than the optimal value. For topology T3 the relative cost oscillates between −75% and −60%.

8.4.2 Runtime

During our simulations, we also measured the mean runtime of the algorithms to evaluate their scalability in terms of computing time. The obtained results are gathered in Table 8. Due to its complexity, the ILP displays high computation times for all of the simulation configurations. For topology T1, the runtime for the ILP starts at 19s for SFC length of 4 and increases to up to 122s for SFC length 10, similar behavior is observed for topologies T2 and T3 with an increase from 24.58s and 24.19s for SFC length 4 to 110s and 102s for SFC length 10 respectively.

The Branch and Bound algorithm turns out to be more costly than the ILP in terms of runtime, indeed, the runtime for longer SFC lengths increases to up to 249s. Overall, the NLPP method is the one with the highest runtime values

SFC Length	4	5	6	7	8	9	10
Topology T1							
ILP	19.36	30.54	44.16	55.13	78.76	82.94	122.26
B&B - LPP	21.02	25.23	47.8	65.67	97.69	149.04	174.67
B&B - NLPP	25.95	33.89	64.5	86.48	118.65	173.86	186.96
B&B - GPP	20.73	25.74	47.82	62.19	86.6	124.81	144.41
H - N1	2.87	2.1	4.2	6.21	6.66	9.11	11.05
H - N2	1.27	1.06	1.76	1.52	2.81	2.75	3.5
H - N3	0.86	0.71	0.93	0.91	0.95	1.41	1.64
Topology T2							
ILP	24.58	33.44	45.97	60.24	72.53	98.24	110.6
B&B - LPP	24.28	51.58	82.21	123.2	146.36	151.73	249.4
B&B - NLPP	27.34	63.15	95.22	129.86	146.12	154.96	218.9
B&B - GPP	24.22	51.74	79.21	112.72	129.12	137.25	205.17
H - N1	2.89	4.49	7.31	6.79	7.54	11.65	11.57
H - N2	1.21	2.64	2.83	2.01	3.29	5.36	4.22
H - N3	0.65	0.95	1.54	1.32	1.86	2.38	2.38
Topology T3							
ILP	24.19	35.29	45.13	56.22	77.96	102.43	102.95
B&B - LPP	42.49	60.92	98.14	143.99	171.21	232.39	234.92
B&B - NLPP	49.61	74.82	118.66	164.38	195.13	241.01	236.18
B&B - GPP	45.2	65.68	102.73	140.63	172.14	210.5	209.1
H - N1	8.09	10.84	13.43	12.4	19.46	16.93	15.84
H - N2	2.24	3.83	5.12	4.58	6.29	3.66	3.83
H - N3	1.73	1.86	1.61	1.85	3.15	2.74	2.24

TABLE 8: Comparison of computation times (in seconds)

starting at 25.95s for SFC length of 4 and increasing to 186s for SFC length 10 when used on topology T1, when for topologies T2 and T3 it increases from 24.58s and 24.19s for SFC length 4 to 110s and 102s for SFC length 10 respectively. The LPP method comes second with a runtime of 21s to 174s for topology T1, and 24s and 42s to 249s and 234s respectively for topologies T2 and T3. Lastly, Global Physical Programming displays runtimes of 20s,24s, and 45s to 144s, 205s, and 209s for topologies T1, T2, and T3 respectively.

In contrast, the heuristic proves to be the most scalable as it outputs results in significantly lower times compared to the exact solutions. The heuristic algorithm with the highest generation number N1 outputs results after 2.1s and up to 11.05s for the topology T1, while for the topologies T2 and T3 the proposed heuristic returns results after 2.89s to 11.65s and 8.09s to 19.46s respectively. Reducing the number of generations to N2 further reduces the runtime, with values of 1.06s to 3.5s for topology T1, and between 1.21s and 5.36s for topology T2, while for topology T3 the runtime ranges between 2.24s and 6.29s. The generation number N3 produces the lowest runtimes with 0.71s to 1.64s, 0.65s to 2.38s, and 1.73s to 2.24s for topologies T1, T2, and T3 respectively. We can therefore conclude that the proposed exact solutions are not scalable, and are not suitable for use at runtime, while the heuristic approach allows results within less than 20 seconds for the biggest instances, and less than a second for the small instances when reducing the number of generations.

9 Conclusion and Future Work

In this work, we proposed a centralized framework to support SFC placement over multiple domains while allowing the domain operators to disclose minimal information on their infrastructure. We modeled the SFC placement problem as a multi-objective optimization problem where the end-to-end latency, individual bandwidth, and relative deployment cost are optimized using Physical Programming to express preferences through meaningful parameters. Three Physical Programming methods were used : Linear, Non Linear, and Global. We implemented our model using an exact algorithm, and also formulated a scalable and efficient heuristic solution. The evaluation of the framework and the proposed algorithms proved our solution's effectiveness with a limited visibility on the network, and the scalability of the heuristic that provided results within the desirable ranges in relatively low runtimes. Furthermore, we were able to demonstrate the benefits of using Physical Programming as opposed to the weighted-sum method by comparing our results to those obtained from an ILP. The comparison between the three Physical Programming methods also provided insights on the results of each one, and we were able to establish that the Linear method slightly outperformed the other ones.

However, the end-to-end life-cycle management of SFCs also requires resource orchestration post-deployment. Indeed, several 5G use cases are characterized by the mobility of users, such as autonomous vehicles, and e-Health. In both use cases, cameras and/or sensors capture information on the environment/patient and send that data to the cloud for processing, and similar to the IIoT use case, security and pre-processing functions are deployed on edge clouds. Since the users are in constant mobility, they might move away from the edge cloud, far enough to increase the latency past the accepted limit. In that case, the operator might have to migrate the edge VNFs to edge clouds that are closer to those users, thus re-computing the SFC placement post-deployment. Furthermore, in the context of fault management, cases of link or node failure would prompt the operator to re-compute the optimal link and node mapping of the affected SFC, and carry out flow re-routing, VNF migration or VNF re-instanciation. In future works, we plan to tackle SFC resource management post-deployment to obtain an orchestration solution that manages SFCs during their entire life-cycle.

References

[1] A. Gupta and R. K. Jha, "A survey of 5g network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

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

[3] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, Apr 2004. [Online]. Available: https://doi.org/10.1007/s00158-003-0368-6

[4] A. Messac, "Physical programming - effective optimization for computational design," *AIAA Journal*, vol. 34, no. 1, pp. 149–158, Jan 1996. [Online]. Available: https://doi.org/10.2514/3.13035

[5] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[6] ETSI, "MEC in 5G networks," ETSI, White Paper (WP) 28, 06 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[7] Azure IoT Edge, https://azure.microsoft.com/en-us/services/iot-edge/.

[8] AWS IoT Greengrass, https://aws.amazon.com/greengrass.

[9] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. Abbas, "An in-depth analysis of iot security requirements, challenges and their countermeasures via software defined security," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[10] T. Kuo, B. Liou, K. C. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM 2016*, April 2016, pp. 1–9.

[11] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining," *IEEE Trans. on Network and Service Management*, vol. 16, no. 1, pp. 374–388, March 2019.

[12] 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.
[13] 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.
[14] A. Abujoda and P. Papadimitriou, "Distnse: Distributed network service embedding across multiple providers," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2016, pp. 1–8.
[15] 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)*, pp. 406–412.
[16] 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.
[17] 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.
[18] 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.
[19] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.
[20] A. Messac, S. Gupta, and B. Akbulut, "Linear physical programming: A new approach to multiple objective optimization," *Transactions on Operational Research*, vol. 8, 01 1996.
[21] J. Sanchis, M. A. Martinez, X. Blasco, and G. Reynoso-Meza, "Modelling preferences in multi-objective engineering design," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 8, pp. 1255 – 1264, 2010.
[22] I. Mehmet Ali and G. Surendra, "Physical programming: A review of the state of the art," *Studies in Informatics and Control*, vol. 21, no. 4, pp. 349–366, 2012.
[23] N. Zhang, "Physical programming based multidisciplinary optimization for aircraft conceptual parameter design," in *2011 Chinese Control and Decision Conference (CCDC)*, May 2011, pp. 2387–2392.
[24] R. Chai, A. Savvaris, A. Tsourdos, and Y. Xia, "An interactive fuzzy physical programming for solving multiobjective skip entry problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 5, pp. 2385–2398, Oct 2017.
[25] H. Li, M. Ma, and W. Zhang, "Multi-objective collaborative optimization using linear physical programming with dynamic weight," *Journal of Mechanical Science and Technology*, vol. 30, no. 2, pp. 763–770, Feb 2016.
[26] S. An, S. Yang, Y. Bai, and X. Wu, "An improved physical programming method for multi-objective inverse problems," in *International Journal of Applied Electromagnetics and Mechanics*, vol. 52, no. 3-4, pp. 1151–1159.
[27] 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.
[28] M. A. Tahmasbi Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vspace: Vnf simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, March 2018.
[29] Linearization of the product of two variables, https://www.leandro-coelho.com/linearization-product-variables/.
[30] 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.
[31] Gurobi, http://www.gurobi.com/.
[32] NetworkX, https://networkx.github.io/.
[33] GSMA, "Network Slicing Use Case Requirements," GSMA, White Paper (WP), 04 2018.
[34] 3GPP, "Technical Specification Group Services and System Aspects;Policy and charging control architecture," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.203, 09 2018, version 15.4.0.
[35] I. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *CoRR*, vol. abs/1708.02562, 2017.
[36] S. Khatibi, "5g mobile network architecture for diverse services, use cases, and applications in 5g and beyond deliverable d6.1 documentation of requirements and kpis and definition of suitable evaluation criteria contractual date of delivery," 09 2017.

Nassima Toumi received her master's degree in Computer Systems Security and her bachelor's degree in Networking and Telecommunications Engineering from the University of Sciences and Technology Houari Boumediene, Algiers, Algeria in 2017 and 2015 respectively. She is currently pursuing her Ph.D. at Sorbonne Universités, Paris, France. She is conducting her Ph.D. research at Orange Labs, Lannion, France and the Communication Systems Department of EURECOM, Sophia Antipolis, France. Her research interests include Service Function Chaining, Software Defined Networking, and Network Function Virtualization.

Olivier J. Bernier received the diploma Ingénieur de l'École Polytechnique from the École Polytechnique, Palaiseau, France in 1986 and the diploma of the École Nationale Supérieure des Télécommunications, Paris, France in 1988. He has been working as a Research Engineer at FTRD, the France Telecom research center, which became Orange Labs, the RD center of Orange, since 1988. His major areas of interest were computer vision, neural networks, and their application to image analysis. He received the HDR ("Habilitation à diriger des recherches") from the Université Pierre et Marie Curie, France in 2009 for his contributions on "Computer vision: face and gestures". Since 2009 his major areas of interest shifted to network monitoring and diagnostic, and the use of AI for network faults detection.

Djamal-Eddine Meddour received his computer engineering degree with honors from the INI (Institut National d'Informatique), Algiers, Algeria, in 2000, his master's degree in computer science from the University of Versailles, France, in 2001 and his Ph.D. in Computer Science from University of Paris VI in 2004. He worked as a senior researcher with Orange Labs Networks (formerly France Telecom RD), and is currently with Orange France as responsible of probes and tool deployment for the fixed network division. His main research activities concern the design and management for light wireless and mobile infrastructure networking, management, and interoperability in new generation wireless networks. He served as a guest editor and TPC member for numerous international conferences and journals. He is the co-author of several international journal articles and book chapters, and holds many patents.

Adlen Ksentini is a COMSOC distinguished lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2016, he worked at the University of Rennes 1 as an assistant professor. During this period, he was a member of the Dionysos Team with INRIA, Rennes. Since March 2016, he has been working as an assistant professor in the Communication Systems Department of EURECOM. He has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking, mobile networks, and more recently on Network Slicing and 5G in the context of H2020 projects 5G!Pagoda, 5GTransformer, 5G!Drones and MonB5G. He has co-authored over 120 technical journal and international conference papers. He received the best paper award from IEEE IWCMC 2016, IEEE ICC 2012, and ACM MSWiM 2005. He has been awarded the 2017 IEEE Comsoc Fred W. Ellersick (best IEEE communications Magazine's paper). Adlen Ksentini has given several tutorials in IEEE international conferences, IEEE Globecom 2015, IEEEE CCNC 2017, IEEE ICC 2017, IEEE/IFIP IM 2017. Adlen Ksentini has been acting as TPC Symposium Chair for IEEE ICC 2016/2017, IEEE GLOBECOM 2017, IEEE Cloudnet 2017 and IEEE 5G Forum 2018. He has been acting as Guest Editor for IEEE Journal of Selected Area on Communication (JSAC) Series on Network Softwerization, IEEE Wireless Communications, IEEE Communications Magazine, and two issues of ComSoc MMTC Letters. He has been on the Technical Program Committees of major IEEE ComSoc, ICC/GLOBECOM, ICME, WCNC, and PIMRC conferences. He acted ss the Director of IEEE ComSoc EMEA region and member of the IEEE Comsoc Board of Governor (2019-2020). He is the chair of the IEEE ComSoc Technical Committee on Software (TCS).

APPENDIX A
PHYSICAL PROGRAMMING FORMULATIONS

A.1 Linear Physical Programming

A.1.1 Interval limits on the y-axis

$$\bar{g}_1 = 0, \quad \bar{g}_2 = \tilde{\bar{g}}^2 \text{ is a small positive number (e.g 0.1)} \quad (1)$$

In order to enforce the OVO rule, the following relationship is applied:

$$\tilde{\bar{g}}^k = \beta(n_{sc} - 1)\tilde{\bar{g}}^{k-1}; (3 \leq k \leq 5); n_{sc} > 1; \beta > 1 \quad (2)$$

Where β is a convexity parameter that needs to be determined.

A.1.2 Convexity

In a contribution from Ma *et al.* [1], a simplified method is proposed for obtaining the appropriate value of β for each criterion in one iteration using the following inequality system:

$$\begin{cases} \beta > 1 \\ \beta > \frac{g_{i,k}}{g_{i,k-1}(n_{sc}-1)}; \quad (3 \leq k \leq 5) \end{cases}$$

Solving this system produces a value for β that satisfies the OVO-rule as well as the convexity requirement.

A.2 Non Linear Physical Programming

In the following, we provide the mathematical formulation of the NLPP class functions, as well as the equation system that allows the computation of the convexity parameter β, more details on the method that was employed to obtain these equations can be found in the original NLPP paper[2].

A.2.1 Interval limits on the y-axis

$$\bar{g}_1 = \tilde{\bar{g}}^1 \text{is a small positive number (e.g 0.1)} \quad (3)$$

$$\tilde{\bar{g}}^k \equiv \bar{g}_i[g_{i(k)}] - \bar{g}_i[g_{i(k-1)}] = \beta n_{sc}\tilde{\bar{g}}^{k-1}, 2 \leq k \leq 5 \quad (4)$$

A.2.2 Convexity

The convexity parameter β must be greater than 1 and ensure that both constants a and b are strictly positive. β is first set at 1.5 then gradually increased by 0.5 until strictly positive values are obtained for a and b for each range of each criterion. Their values are calculated as follows :

$$a = \frac{3[3s_{i,k}+s_{i,k-1}]-12\tilde{s}_i^k}{2(\lambda_i^k)^3} \quad (5)$$

$$b = \frac{12\tilde{s}_i^k-3[s_{i,k}+3s_{i,k-1}]}{2(\lambda_i^k)^3} \quad (6)$$

Where $\tilde{s}_i^k$ is a characteristic slope for the kth region of the criterion i as expressed in Eq 7. And s_i^k allows the computation of the slope of a region according to the slope of the preceding region as is shown in Eqs 8-9.

$$\tilde{s}_i^k = \tilde{\bar{g}}^k / \lambda_i^k \quad (7)$$

$$s_{i1} = \alpha \cdot \tilde{s}_i^2, 0 < \alpha < 1 \quad (8)$$

$$s_{i,k} = \frac{(8\alpha+4)\tilde{s}_i^k-(8\alpha+1)s_{i,k-1}}{3}, 2 \leq k \leq 5 \quad (9)$$

Note that the constant α should be kept at a relatively small value (<0.1) in order to maximize the range of acceptable slopes at the region boundaries.

A.2.3 Class Functions

For each objective *i*: For the soft class 1-S, the following model is given:

– For the first range ($(g_i \leq g_{i,1})$):

$$\bar{g}_i = \bar{g}_1 \cdot e^{\left[(s_{i,1}/g_{i,1})(g_i-g_{i,1})\right]} \quad (10)$$

– For each other range:

$$\bar{g}_i = T_0(\xi_i^k)\bar{g}_{k-1} + T_1(\xi_i^k)\bar{g}_k + \bar{T}_0(\xi_i^k;\lambda_i^k)s_{i,k-1} + \bar{T}_1(\xi_i^k;\lambda_i^k)s_{ik} \quad (11)$$

$$\xi_i^k = \frac{g_i-g_{i,k-1}}{\lambda_{i,k}} \quad (12)$$

Where $0 \leq \xi_i^k \leq 1, k = 2, ..., 5$, and

$$T_0(\xi) \equiv \frac{1}{2}\xi^4 - \frac{1}{2}(\xi-1)^4 - 2\xi + \frac{3}{2} \quad (13)$$

$$T_1(\xi) \equiv -\frac{1}{2}\xi^4 + \frac{1}{2}(\xi-1)^4 + 2\xi - \frac{1}{2} \quad (14)$$

$$\bar{T}_0(\xi,\lambda) \equiv \lambda[\frac{1}{8}\xi^4 - \frac{3}{8}(\xi-1)^4 - \frac{1}{2}\xi + \frac{3}{8}] \quad (15)$$

$$\bar{T}_1(\xi,\lambda) \equiv \lambda[\frac{3}{8}\xi^4 - \frac{1}{8}(\xi-1)^4 - \frac{1}{2}\xi + \frac{1}{8}] \quad (16)$$

A.3 Global Physical Programming

A.3.1 Interval limits on the y-axis

The images $\bar{g}^k$ at the range boundaries g_i^k are calculated as follows :

$$\bar{g}_0 = 0, \bar{g}_1 = \bar{g}^{ini}, \bar{g}^{ini} \geq 0 \quad (17)$$

$$\bar{g}^k = \bar{g}^{k-1} * n_{sc} (2 \leq k \leq N) \quad (18)$$

With N being the number of preference ranges, and $\bar{g}^{ini}$ a pre-defined parameter.

1. X. Ma and B. Dong, "Linear Physical Programming-Based Approach for Web Service Selection," 2008 International Conference on Information Management, Innovation Management and Industrial Engineering, Taipei, 2008, pp. 398-401, doi: 10.1109/ICIII.2008.156.

2. A. Messac, "Physical programming - effective optimization for computational design,"AIAA Journal, vol. 34, no. 1, pp. 149–158,Jan 1996. [Online]. Available: https://doi.org/10.2514/3.1303