A multi-stage graph aided algorithm for distributed Service Function Chain provisioning across multiple domains

GODFREY KIBALYA[1], JOAN SERRAT[1], JUAN-LUIS GORRICHO[1], DOREEN BUJJINGO [2] AND JONATHAN SERUGUNDA[2]

[1]Dept. Network Engineering, Universitat Politecnica de Catalunya, Barcelona - Spain (e-mail: Godfrey.mirondo.kibalya@upc.edu, serrat@tsc.upc.edu, juanluis@entel.upc.edu)

[2]Dept. Electrical and Computer Engineering, Makerere University, Kampala- Uganda (e-mail: doreenserene04@gmail.com,serugthan@gmail.com)

Corresponding author: Godfrey Kibalya (e-mail: Godfrey.mirondo.kibalya@upc.edu).

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777067 (NECOS project) and the national project TEC2015-71329-C2-2-R (MINECO/FEDER). This work is also supported by the "Fundamental Research Funds for the Central Universities" of China University of Petroleum (East China) under Grant 18CX02139A

ABSTRACT Network Service Providers (NSPs) envisage to support the divergent and stringent requirements of future services by instantiating these services as service chains, commonly referred to as Service Function Chains (SFCs), that are customized and configured to meet specific service requirements. However, due to the limited footprint of the Infrastructure Providers (InPs), these SFCs may have to transcend multiple InPs/domains. In this regard, determining the optimal set of InPs in which to embed the SFC request emerges as a complex problem for several reasons. First, the large number of possible combinations for selecting the InPs to embed the different sub-chains of the request makes this problem computationally complex, rendering optimal solutions only after long computations, especially in large scale networks, which is unfeasible for delay sensitive applications. Second, the unwillingness of InPs to disclose their internal information, which may be vital for making embedding decisions, usually implies the provisioning of single-domain solutions, which are unsuitable in this working scenario. In this regard, this paper first formulates the multi-domain service deployment problem under multiple request constraints, such as bandwidth or delay, among others. Then, due to the NP-hardness nature of the above problem, this paper proposes an algorithm that is aided by a multi-stage graph for computing a request embedding solution in a distributed manner, solving the problem in acceptable run-times. Results from different simulations reveal that the proposed algorithm is optimized in terms of acceptance ratio and embedding cost, with up to 60.0% and 88.7% improvements in terms of embedding cost and execution time, respectively, for some scenarios, in comparison with a benchmark state-of-the-art algorithm.

INDEX TERMS Service Function Chaining, Distributed Algorithm, Multi-domain Embedding, Network Function Virtualization.

I. INTRODUCTION

Network Function Virtualization offers great prospects for building logical networks with the ability to support the divergent and stringent requirements of future services through the softwarization of the network functions, hitherto implemented by middle-boxes coupled to proprietary hardware [1], [2]. By this approach, NFV provides the possibility of migrating those network functions from dedicated hardware appliances to general purpose computing, storage, and networking solutions [3], [4]. This facilitates a dynamic network management by allowing multiple service-specific virtual networks to be deployed on a shared infrastructure [5], [6]. In this regard, end-to-end services will be instantiated as service chains consisting of an ordered set of Virtual Network Functions (VNFs), commonly refereed to as Service Function Chains (SFCs), which can be easily and dynamically deployed, scaled or migrated [7]–[9].

The problem of service deployment considering a single administrative network infrastructure has been extensively addressed in the literature [10]–[14]. However, for

a number of practical scenarios such as Internet-of-Things applications, the geographical location at which the data is generated may be different from where it will be accessed, processed or consumed [15], [16]. This, coupled with the location dependencies of certain network functions, and the limited footprint of InPs, may necessitate the SFCs to transcend multiple domains, and then, involve several infrastructure providers [17], [18]. Therefore, the end-to-end service supporting such applications must be materialized by a chain of service instances supported by different InPs. This brings extra complexity regarding how to efficiently deploy the different service chains onto an underlying infrastructure belonging to multiple providers, while meeting the stringent constraints associated with the requests in terms of both required amount of resources and end-to-end delay. The problem is further exacerbated by the unwillingness of the different InPs to disclose information related to their internal network topology, although, that information would be vital to make efficient mapping decisions [19]. Fig. 1 shows an example of several IoT systems in which information from medical and road traffic domains are relayed to a core data-center (with the possibility that there are multiple core cloud servers belonging to different InPs) through several access networks (wireless and optical) comprised of multiple InPs.

Algorithms for a multi-domain service deployment are either centralized or distributed: centralized algorithms rely on a centralized entity that uses global information to make decisions about the different InPs for hosting the requests. This may affect the scalability of such approaches when considering large scale networks. Moreover, different entities cannot compute mapping solutions in parallel, affecting the execution time of the algorithm [20]. On the other hand, deciding the placement of any request when using distributed algorithms involves the participation of different InPs. Unlike centralized algorithms, as discussed in [21], distributed algorithms are well suited when considering dynamic network environments, and they are also suited to protect the privacy of the different InPs, hence, working with limited information disclosure as considered in this work. However, distributed approaches are penalized by an increasing processing delay and signaling overhead when making request provisioning decisions as the number of participating InPs increases, hence, compromising their scalability. To overcome this challenge, this paper proposes an algorithm for the distributed provisioning of service requests across multiple infrastructure providers which incorporates two innovative features: first, unlike other distributed algorithms, such as in [22], where a messages exchange occurs between any node and all its neighbors, the messages exchange in our work involves only a pre-computed set of candidate nodes, thanks to the use of a candidate extraction step; secondly, we incorporate a message processing technique in which, upon receiving message blocks from other InPs, each candidate InP processes the received messages and forwards only a single message, based on all the received messages, to a given sub-set of InPs. This, significantly reduces the

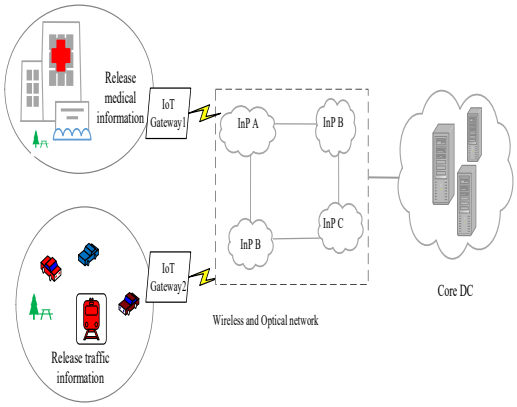


FIGURE 1: An illustration of multi-domain service deployment

computational overhead of the different InPs participating in the solution to be computed. Moreover, it is possible to detect unfeasible requests in early stages when running our proposed algorithm, further enhancing the algorithm time performance, hence, rendering a well suited approach for delay sensitive applications. Moreover, different from existing works addressing the multi-domain service problem, our work incorporates multiple intra-domain performance parameters, such as processing costs, intra-domain delays, VNF activation costs or energy costs, among others.

Therefore, our contributions in this work can be summarized as follows:

1) An algorithm for the provisioning of service requests in a distributed manner across an underlying infrastructure belonging to multiple InPs. The computation of the solution is based on the use of a multi-stage graph composed of a sub-set of InPs, enhancing its execution time with no degradation in its performance.
2) A formal description and formulation of the SFC placement problem across a multi-provider network infrastructure while satisfying end-to-end delay constraints.
3) A candidate selection algorithm to pre-compute the set of candidate InPs to participate in the solution computation of the embedding algorithm. This minimizes the number of InPs participating in the computation of the embedding solution.
4) Extensive simulations considering both offline and online scenarios. From the simulation results, the proposed algorithm is found to be scalable when increasing both the network size and the request demand. Moreover, the proposed algorithm is also found to be optimal in terms of some selected performance metrics, including the provisioning cost and the execution time, compared to a state of the art benchmark algorithm.

The rest of the paper is organized as follows: section II presents the related work. The network modeling and problem description is presented in section III. The proposed

multi-stage graph based distributed algorithm is described in section IV. The performance evaluation of the proposed algorithm is presented in section V. Finally, the conclusions are summarized in section VI.

A. SOLUTION CONTEXT

The proposed solution is aligned with the "NFV Infrastructure as a service (NFVIaaS)" use case as described in [23], and the "Network Services provided using multiple administrative domains" use case described in [24], both of them applicable to scenarios where a single service provider is unable to meet the requirements of its consumers. This is a realistic scenario since, in practice, consumers may demand services with a global span, yet, many service providers may not have the capacity to deploy and provide resources around the globe due to different reasons, including financial or regulatory constraints, among others. Under such perspective, an effective approach for any service provider comes from using resources of different infrastructure providers. As articulated in [24], under the NFVIaaS use case, the tasks of: the VNF placement decision, the management of software images for the deployed VNFs, the SLA supervision or the management of the intra-domain VNF infrastructure, among others, are delegated to the NFVIaaS provider, with whom the NFVIaaS consumer establishes an "a priori" commercial agreement. This justifies our distributed approach for the targeted problem, since in practice, a given InP will have limited control and visibility of the network operations happening in another InP domain.

The different architecture options, through which the logical interconnection and service orchestration in a multi-provider scenario can be supported, are proposed and described in the ETSI report [24]. The ETSI NFV-MANO architectural framework described in [25], and shown in Fig. 2, serves as the basis for the aforementioned multi-domain architecture options, with additional enhancements of the interfaces and reference points where necessary, depending on the specific architecture option. The ETSI NFV-MANO architectural framework is constituted of a set of functional blocks, data repositories used by these blocks, and the respective interfaces and reference points through which the different blocks can exchange information in order to effectively manage the virtualized infrastructure and the corresponding services within a given administrative domain. The key building blocks of the architecture are: the NFV Orchestrator (NFVO), the Virtualized Infrastructure Managers (VIMs) and the VNF Manager (VNFM). The NFVO is responsible for the orchestration of NFVI resources across multiple VIMs and the life-cycle management of the deployed network services. The VNFM is responsible for the life-cycle management of VNF instances, including VNF instantiation, modification, healing and termination, among others. On the other hand, the VIM is in charge of controlling and managing the NFVI compute, storage and network resources within a given domain. In order to achieve a multi-domain connectivity, the architecture options permit the exchange of information among different domains, including IP addresses of the distinct functional blocks to be interconnected, such as the NFVO, the unique identifiers of the administrative domains to be interconnected, and the administrative organization they pertain to, among others. Moreover, the proposed architecture options may allow for auto-discovery mechanisms in which the different NFV-MANO functional blocks of the different domains can advertise their own information which can be exploited by the discovery mechanisms to establish a connectivity relation [24].

The architecture option for the "Network Services provided using multiple administrative domains" use case is shown in Fig. 3, which considers a case where there is a single NFV Orchestrator (NFVO) per administrative domain. In this case, a new reference point $Or - Or$ is proposed to be added to the NFV-MANO architecture to facilitate the communication between the different NFVOs in order to enable a life-cycle management of the deployed composite service. In the shown architecture example, NFVO-1 (which we denote as the master orchestrator in our work) is in charge of the life-cycle management of the composite service, including initiation of scaling operations when necessary, while NFVO-2 and NFVO-3 are responsible for the life-cycle management of the nested services (NSs) running inside their respective administrative domains. However, NFVO-1 is unaware of the virtualized resources in the host domains of both NFVO-2 and NFVO-3, with the interaction between the VNFM of each domain being limited to the respective NFVO of that domain. In this regard, a service deployment algorithm that is cognizant of the limited information exposure is well suited for service deployment under this scenario. Moreover, abstracting the internal topology of the different domains from the global orchestrator has been found to result in a significant reduction in the solution computation time with a tenable cost increment [18], [26], [27]. The authors in [26] and [18], [27] analyzed the time reduction gain and provisioning cost performance, respectively, resulting from abstracting the internal topologies of the different domains in the multi-provider service deployment problem.

II. RELATED WORK

There are significant contributions in the literature devoted to solving the problem of the service embedding for single domain environments; such as in [3], [28]–[31]. However, such approaches rely on a full network topology exposure, which is not a realistic consideration when extended to the multi-domain scenario. The multi-domain service deployment problem has been addressed in the literature either as a Virtual Network Embedding (VNE) problem or as a Service Function Chain Placement (SFC) problem. We summarize here the most representative works in both of these areas.

The authors in [17], [18], [27], [32], [33] adopt approaches based on obtaining exact solutions for the multi-domain service embedding. However, although these ap-

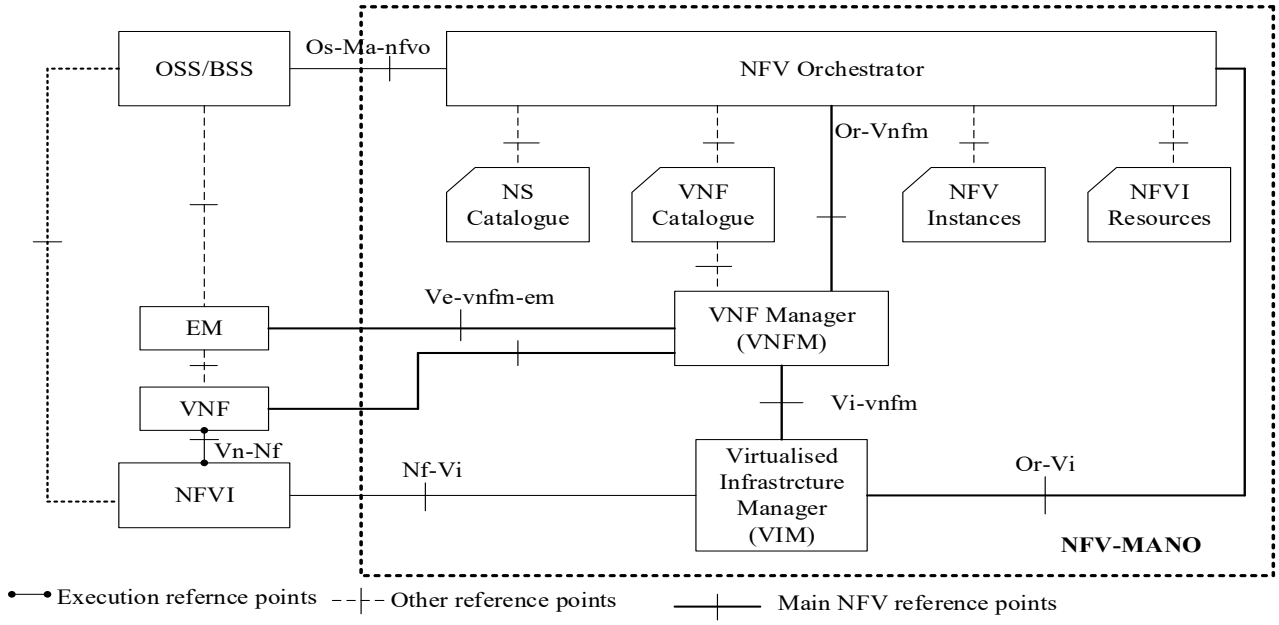


FIGURE 2: NFV MANO architecture

proaches result in optimal solutions, they achieve that at the expense of an increased processing time complexity, hence, they are not well suited for delay sensitive applications as envisaged in future networks, including 5G. Due to this challenge, a number of alternative heuristic approaches have been proposed in order to embed services across multiple domains within a feasible run-time. In [21], [34]–[36] several multi-domain algorithms are proposed considering that the internal information within each InP is visible from other InPs. However, in practice, due to security and business competition issues, InPs are reluctant to share their topological and internal policy information, hence, these approaches are unsuitable for scenarios with restricted information disclosure. The works in [5], [22], [37]–[39] adopt heuristic approaches to map services across multiple domains under limited information disclosure. In [5], the algorithm uses the exposed boarder nodes to compute all the feasible paths from source to destination. Then, for each of these paths, the first InP on the path receives the SFC request and selects a sub-SFC to bid for, and forwards this and the SFC to the next InP along the path. Then, the receiving InP also selects a sub-SFC among the non-selected VNFs and also tries to compete for the sub-SFC selected by the previous InP. This process continues until the last InP along the path selects or competes for a sub-SFC of the request. Then, the path for mapping the request is chosen as the one which results in the least cost among all candidate paths from source to destination. The work in [38] adopts a similar approach in which the exposed boarder nodes are used to obtain feasible abstracted paths connecting the source node with the destination node. The algorithm, then, partitions the SFC according to two criteria, namely, according to the number of domains crossed by the abstracted path, with the goal of minimizing the end-to-end delay, and according to the available physical resources of the different domains constituting the abstracted paths. The authors of the above work adopt a similar approach in [40] and [39], with the added goal of minimizing the energy consumption. However, as revealed from the simulation results of this paper, using the exposed boarder nodes to compute all possible paths from source to destination has a high time complexity, which greatly affects the running time of the entire algorithm. The work in [37] introduces pSMART, a privacy-aware SFC approach that targets to jointly achieve privacy and a high multi-domain SFC orchestration efficiency. However, like most existing works, the work adopts a centralised approach in which the solution is evaluated based on computing the K shortest paths between the ingress and egress nodes, using an abstracted topology. However, in practice, the provisioning cost may be affected by multiple intra-domain undisclosed costs, including energy, processing, link and QoS violations, among others. In this way, the shortest path may not necessarily result in the least provisioning cost. Moreover, obtaining a near-optimal solution may require obtaining a large number of paths, which compromises the execution time of the algorithm.

The work in [41] proposes a service deployment framework which incorporates innovative features, such as: resource availability prediction and incremental learning, among others, in order to realize a service deployment solution that is adaptive to temporal variations in network or request requirements states. In [42] the focus is set on scheduling micro-services on multiple clouds, including micro-clouds, that could belong to a single or multiple

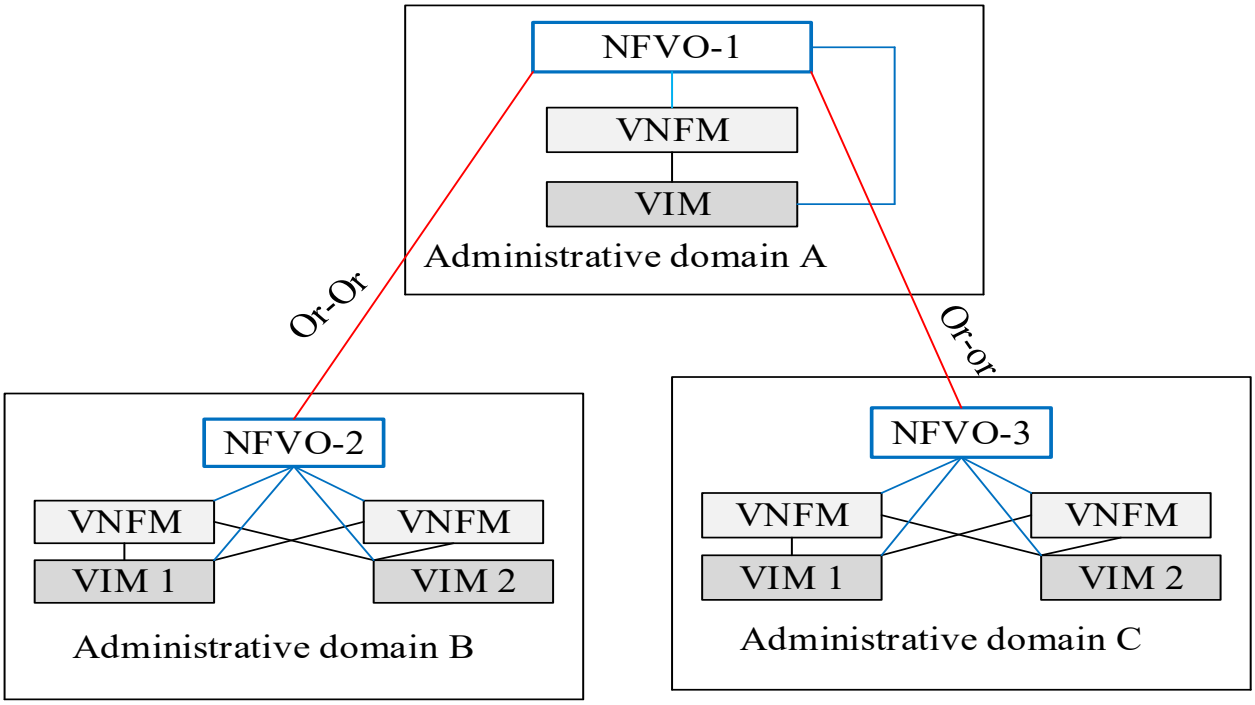


FIGURE 3: ETSI proposed architecture option for Network Services provided using multiple administrative domains use case [24]. In this illustration, domain A is the originating domain of the service request with NFVO-1 being the master orchestrator (MO) in charge of life-cycle management of the composite service. NFVO 2 and 3 are restricted to the life-cycle management of the nested services inside their respective domains

operators, using a fair weighted affinity-based s cheme to solve the scheduling problem. However, these works rely on a full knowledge of the resources used within the different clouds, which may not be accessible under partial information disclosure. While considering a distributed cloud environment under a multi-stakeholder setting, the authors in [43] target to optimise the utility of users, Service Providers and the Infrastructure Providers by adopting an ILP formulation that leverages a multi-layered auxiliary graph built for each request to be provisioned.

In [22] a distributed algorithm is proposed in such a way that, upon the arrival of a request, the centralized orchestrator forwards the request to the different participating InPs. Then, following their internal policies, each InP selects the sub-SFC it can map. All the intra-domain mappings are then forwarded to the orchestrator, and this one will select the optimal InPs for hosting the request with the goal of minimizing the overall provisioning cost. However, during the distributed computation, the algorithm requires the different InPs to forward signaling requests to all their reachable neighbors, increasing the time for making a mapping decision, especially as the number of InPs increases. In [44], a distributed embedding algorithm is proposed for the single VNE problem. In this case, each node behaves as an autonomic agent. However, the messages exchange overhead is unavoidable as the number of substrate nodes increases, since, even the unfeasible nodes participate in the solution computation. Moreover, the algorithm execution entails a path computation step which involves computing paths between all nodes, which is very time consuming.

In [2], [26], [45], [46] different multi-stage based approaches are proposed for solving the service embedding problem. The embedding solution in these works is obtained by either applying the Viterbi-algorithm [45], [46], or a flow based algorithm [2], [26], directly on the graph. However, all these approaches require a centralized entity which has a global view of the weights of the nodes and links constituting the graph, something that is not feasible under a scenario of partial information disclosure regarding those node and link weights. Moreover, different from these approaches, the multi-stage graph tool in our work is only used to establish neighborhood relationships between the different candidates, and it is not used for directly computing the mapping solution.

III. SERVICE AND NETWORK MODELING - PROBLEM DESCRIPTION

In this section we describe the mathematical modeling of the SFC requests and the substrate network providing the service. In addition, a description of the multi-domain service embedding problem is given, also from a mathematical perspective.

TABLE 1: Notations and variables

Notation	Description
K	Number of InPs in the substrate network
$\mathbb{K}$	A set of all InPs
G_s	Substrate network topology graph
G_s^k	Substrate network topology graph for domain/InP k
N_s, E_s	Set of all substrate nodes and edges respectively
N_s^k, E_s^k	Set of all substrate nodes and edges respectively for InP k
Ψ^r	Tuple of all request attributes
n_s^k	Substrate node in domain K
x_n^s, y_n^s	x and y coordinate respectively for node n^s
$f^{n_s^k}$	Set of function types that can be deployed on n_s^k
$c_{res}^{n_s^k}$	Amount of residual resources on n_s^k
$C_{max}^{n_s^k}$	CPU capacity of n_s^k
$\zeta_{n_s}^k$	Cost for processing a packet rate unit on n_s^k
e^k	A physical link inside domain k
E_{int}	Set of all inter-domain links
$B_{max}^{e^k}$	Bandwidth capacity of edge e^k
$Bw_{res}^{e^k}$	Residual bandwidth on edge e^k
δ^{e^k}	propagation delay on e^k
ζ^{e^k}	Cost for transmitting a packet rate unit on e^k
G_v^r	Set of ordered VNFs that request $r \in R$ must traverse
n_v^p	Request virtual node of type p
P	Set of all function types
ρ^r	Requested packet rate of request $r \in R$
τ_s^r, τ_d^r	Ingress and egress node respectively for $r \in R$
τ_f^r	Life-time of $r \in R$
$Req_{N_v}^r$	Set containing requirements of the virtual nodes of $r \in R$
$C_{dem}^{n_v^p,r}$	CPU resource requirement of node n_v^p of request $r \in R$
$r^{n_v^p}$	Acceptable location region of $n_v^p \in N_v$ of request
$\rho_{n_v^p}^r$	Packet rate traversing n_v^p
C_ρ	Amount of CPU for processing each unit of packet rate
Del_{sd}^r	Delay requirement for $r \in R$
l_{uv}	Request virtual link between virtual nodes u and v
$Bw_{l_{uv}}^r$	Bandwidth requirement for l_{uv}
R_A	Set of all admitted requests
$\sigma_{l_{uv}}^e \in \{0,1\}$	$\sigma_{l_{uv}}^e$=1 if l_{uv} is provisioned by $e \in E_s$, zero otherwise
$y_{n_s^k}^{n_v^p} \in \{0,1\}$	$y_{n_s^k}^{n_v^p}$=1 if virtual node n_v^p is provisioned by substrate node n_s^k,zero otherwise.
$\gamma_p^{n_s^k} \in \{0,1\}$	$\gamma_p^{n_s^k}$=1 if VNF m of type p has been provisioned by substrate node n_s^k,zero otherwise.
$e_{n_s^k}^{idle}$, $e_{n_s^k}^{busy}$	Idle and peak power consumption of node n_s^k
δ_{vnf}^p	Processing delay at VNF of type p
Υ_{vnf}^p	Set of all substrate nodes that can provision a VNF of type p
$z_{mp} \in \{0,1\}$	$z_{mp}^p = 1$ if VNF m is of type P, zero otherwise.
$Cand_s^r$	Set of candidate InPs for request $r \in R$
$Cand^{n_v^p}$	Set of candidate InPs for virtual node n_v^p
$dist(n^s, i)$	Euclidean distance between substrate node n^s and virtual node $loc(i)$

A. SFC REQUEST

Considering a set of R requests, each request $r \in R$ is modeled as a tuple $\Psi^r = < G_v^r, Req_{N_v}^r, \rho^r, Del_{sd}^r, \tau_s^r, \tau_d^r, \tau_f^r >$ where G_v^r is the SFC graph of the VNFs the user traffic must traverse, including the virtual links interconnecting those VNFs. We refer to each of such required VNFs as a request virtual node or virtual node for convenience, denoted by $n_v^p \in N_v$, where N_v denotes the set of all such nodes, and $p \in P$ denotes the function type (e.g. firewall or NAT, among others) of this node. The parameter ρ^r is used to denote the requested packet rate of the user input traffic from the ingress node τ_s^r to the egress node τ_d^r. $Req_{N_v}^r$ denotes a set capturing the requirements of the different request virtual nodes in terms of CPU resources and location constraints, with $C_{dem}^{n_v^p,r}$ and $r^{n_v^p}$ denoting the CPU resource requirement and acceptable location region of $n_v^p \in N_v$, respectively. In practice, the amount of CPU resources required by a node n_v^p is proportional to the packet rate to be processed by that node, i.e. $C_{dem}^{n_v^p,r} = \rho_{n_v^p}^r \times C_\rho$, where $C_{dem}^{n_v^p,r}$ is the amount of CPU resources required by n_v^p, with $\rho_{n_v^p}^r$ and C_ρ denoting the packet rate traversing n_v^p and the amount of CPU resources required to process each unit of packet

rate by that node, respectively. The terms Del^r_{sd}, τ^r_s, τ^r_d, τ^r_f are used to denote the end-to-end delay requirement of the request, the ingress node, the egress node, and the request life-time, respectively. Similarly, we denote by $l_{uv} \in L_v$ the request virtual link between request virtual nodes u and v, and we denote the bandwidth requirement for such a link by $Bw^r_{l_{uv}}$. Note that the amount of required bandwidth may vary across different links, since the packet rate may be altered by the traversed virtual nodes, for instance, as a result of filtering or splitting of packets due to applying some kind of networking functionality.

B. SUBSTRATE NETWORK

The substrate network considered in this work consists of K InPs modeled as a weighted undirected graph $G_s = (N_s, E_s)$ where N_s, E_s denote the set of all physical nodes (e.g. servers) and physical links, respectively. The substrate network of a given domain $k \in \mathbb{K}$ is modeled as a weighted undirected graph $G^k_s = (N^k_s, E^k_s)$, where N^k_s and E^k_s denote the set of substrate nodes and intra-domain substrate links within that domain, where $G^k_s \in G_s$, $N^k_s \in N_s$ and $E^k_s \in E_s$. Each physical node $n^k_s \in N^k_s$ in the k^{th} domain is characterized by: i) a location specification $loc^{n^k_s}$, modeled as a point $p(x^s_n, y^s_n)$, where x^s_n and y^s_n are the x and y Cartesian coordinates; ii) a set of function types that can be deployed onto this node, denoted as $f^{n^k_s}$; iii) its residual CPU resources at a given time, denoted by $c^{n^k_s}_{res}$; iv) a CPU resource capacity, denoted by $C^{n^k_s}_{max}$; and finally, v) the cost of processing each unit of packet rate at this node, denoted by $\zeta^k_{n_s}$. Similarly, we denote each physical link by $e^k \in E^k_s$ within domain k. We also use $e_{int} \in E_{int}$ to denote an inter-domain link, where $E_{int} \subset E_s$ denotes the set of all inter-domain links. Each link $e^k \in E^k_s$ or $e_{int} \in E_{int}$ is characterized by: i) a bandwidth capacity $B^{e^k}_{max}$ or $B^{e_{int}}_{max}$; ii) a residual bandwidth at a given time, denoted by $Bw^{e^k}_{res}$ or $Bw^{e_{int}}_{res}$; iii) a propagation delay δ^{e^k} or $\delta^{e_{int}}$, and iv) a cost for transmitting a packet rate unit ζ^{e^k} or $\zeta^{e_{int}}$.

C. MULTI-DOMAIN SFC PROVISIONING PROBLEM - DESCRIPTION AND FORMULATION

Given a service request to be provisioned, and an underlying substrate network owned by multiple InPs whose internal network topology and pricing information is considered confidential, our objective is to obtain a set of InPs that satisfies the request requirements while resulting in the least request provisioning cost. In a general sense, this problem can be decomposed into three main sub-tasks: the candidates search, the request splitting and the request binding. The candidates search task identifies a set of InPs that can potentially serve that request, either partially or in full, by exploiting the public information disclosed by all the InPs and the requirements of the request. In this regard, this paper proposes a Candidate InPs Identification Algorithm (CIIA) that performs this task. Since this task may associate each request virtual node with more than one possible candidate InP, the request splitting task focuses on selecting a sub-set of feasible InPs, among all possible candidates, in order to optimize the mapping objective, e.g. the cost. In this paper, such a task is implemented by a messages exchange among the candidate InPs obtained from the first task. Once the optimal InPs for embedding the request are identified, the binding task carries out the reservation and allocation of the necessary intra-domain and inter-domain resources along the selected InPs in order to instantiate the end-to-end service. In this work, the target of the service provisioning algorithm is to minimize the average provisioning cost for each admitted SFC request. Moreover, we consider a scenario where the cost for each unit of any node or link resource may vary across different InPs. In principle, minimizing the provisioning cost of any request minimizes the operational cost of the service provider and maximizes the resultant net revenue. We consider the provisioning cost of any request to be influenced by: i) the energy consumption cost associated with running the different VNFs onto the substrate nodes of the different domains; ii) the transmission cost of transferring the user traffic from the ingress node to the egress node along all the intermediate links; and iii) the processing cost incurred for processing the user traffic at the different VNFs traversed. Note, however, that other cost components, such as VNF instantiation, could be easily integrated into the adopted cost model of the algorithm. Therefore, the multi-domain service provisioning problem target can be expressed as:

$$Minimize \quad C(Gv) \tag{1}$$

where $C(Gv)$ denotes the average provisioning cost per admitted SFC request, defined as follows:

$$C(Gv) = \frac{1}{|R_A|} \sum_{r \in R_A} C^r_p(Gv) \tag{2}$$

where $C^r_p(Gv)$ is the provisioning cost for a request $r \in R_A$, and R_A denotes the set of all admitted requests, with $|R_A|$ being the cardinality of that set. In order to evaluate $C^r_p(Gv)$, we define the following variables: $\sigma^e_{l_{uv}} \in \{0,1\}$ is a binary variable, equal to 1 if the request virtual link l_{uv} is provisioned by the intra-domain edge $e \in E^k_s$ of domain $k \in \mathbb{K}$, zero otherwise; $\sigma^{e_{int}}_{l_{uv}} \in \{0,1\}$ is equal to 1 if the request virtual link l_{uv} is provisioned by the inter-domain edge $e_{int} \in E_{int}$, zero otherwise; $y^{n^p_v}_{n^k_s} \in \{0,1\}$ is equal to 1 if the request virtual node n^p_v is provisioned onto substrate node n^k_s, zero otherwise. Then, the request provisioning cost can be evaluated as:

$$\begin{aligned} C^r_p(Gv) = & \sum_{l_{uv} \in L_v} \sum_{k \in \mathbb{K}} \sum_{e^k \in E^k_s} \sigma^{e^k}_{l_{uv}} \times \zeta^{e^k} \times \rho^r \\ & + \sum_{l_{uv} \in L_v} \sum_{e_{int} \in E_{int}} \sigma^{e_{int}}_{l_{uv}} \times \zeta^{e_{int}} \times \rho^r + \\ & \sum_{n^p_v \in N_v} \sum_{k \in \mathbb{K}} y^{n^p_v}_{n^k_s} \times \zeta^k_{n_s} \times \rho^r + \chi_w \sum_{k \in \mathbb{K}} \sum_{n^k_s \in N^k_s} E_{n^k_s} \end{aligned} \tag{3}$$

where the first and second terms of Eqn. 3 correspond to the transmission costs due to the use of the intra-domain and inter-domain edges, respectively, and the third and fourth terms correspond to the processing costs, due to the use of the selected substrate nodes, and the energy costs, respectively. The parameter $E_{n_s^k}$ from the energy cost term denotes the energy consumption at node n_s^k, and χ_w denotes the cost per unit of energy consumption. $E_{n_s^k}$ is computed using the model adopted in [47] as follows:

$$E_{n_s^k} = e^{idle}_{n_s^k} + [e^{busy}_{n_s^k} - e^{idle}_{n_s^k}] \times U^{n_s^k}_{til} \tag{4}$$

where $e^{idle}_{n_s^k}$, $e^{busy}_{n_s^k}$ denote the idle and peak power consumption of the node n_s^k. The term $U^{n_s^k}_{til}$ refers to the utilization of substrate node n_s^k.

In order to increase competitiveness, the mapping inside each domain is done with the objective of minimizing the mapping cost for the sub-SFC that is bid for by the corresponding domain. Complementary, the optimization criterion expressed in Eqn. 1 should adhere to a number of constraints, including the following:

• The total bandwidth consumption on a given edge $e^k \in E_s^k$ and $e_{int} \in E_{int}$ should not exceed the capacity of that edge:

$$\sum_{r \in R} \sum_{l_{uv} \in L_v} \sigma^{e^k}_{l_{uv}} \times Bw^r_{l_{uv}} \leq B^{e^k}_{max} \; \forall e^k \in E_s^K \tag{5}$$

$$\sum_{r \in R} \sum_{l_{uv} \in L_v} \sigma^{e_{int}}_{l_{uv}} \times Bw^r_{l_{uv}} \leq B^{e_{int}}_{max} \; \forall e_{int} \in E_{int} \tag{6}$$

• The end-to-end delay should not exceed the acceptable delay of the request:

$$\sum_{l_{uv} \in L_v} \sum_{k \in \mathbb{K}} \sum_{e^k \in E_s^k} \sigma^{e^k}_{l_{uv}} \delta^{e^k} + \sum_{l_{uv} \in L_v} \sum_{e_{int} \in E_{int}} \sigma^{e_{int}}_{l_{uv}} \delta^{e_{int}} + \sum_{n_v^p \in N_v} y^{n_v^p}_{n_s^k} \times \delta^p_{vnf} \leq Del^r_{sd} \;\; \forall r \in R \tag{7}$$

where δ^p_{vnf} denotes the processing delay experienced by a packet at a VNF of type p. The first and second terms of equation 7 correspond to the propagation delay of the intra-domain and inter-domain edges, respectively, and the third term corresponds to the processing delay at the different VNFs being traversed by the user traffic.

• The CPU consumption at a given substrate node should not exceed the node resource capacity:

$$\sum_{r \in R} \sum_{n_v^p \in N_v} y^{n_v^p}_{n_s^k} \times C^{n_v^p, r}_{dem} \leq C^{n_s^k}_{max} \;\; \forall n_s^k \in N_s^k, k \in \mathbb{K} \tag{8}$$

• Each request virtual node must be mapped onto a single substrate node:

$$\sum_{k \in \mathbb{K}} \sum_{n_s^k \in N_s^k} y^{n_v^p}_{n_s^k} = 1 \;\; \forall n_v^p \in N_v \tag{9}$$

• Each request virtual node should be provisioned on a substrate node that is within its acceptable geographical location:

$$dist(n_s^k, n_v^p) \leq dev(n_v^p) \; \forall \; n_v^p \in N_v \tag{10}$$

where $dist(n_s^k, r^{n_v^p})$ denotes the distance of substrate node n_s^k from the desired location of virtual node n_v^p and $dev(n_v^p)$ denotes the maximum acceptable deviation from such a location.

• Each VNF of type p should be provisioned on a substrate node capable of supporting that type of VNF:

$$\gamma_p^{n_s^k} = 1 \; iff \;\; n_s^k \in \Upsilon^p_{vnf} \;\; \forall \, p \in P \tag{11}$$

where Υ^p_{vnf} is a set containing all nodes that can provision a VNF of type p.

• Similarly, a request virtual node n_v^p is provisioned by substrate node n_s^k only if there is a VNF of type p already provisioned on that node.

$$y^{n_v^p}_{n_s^k} = min\{y^{n_v^p}_{n_s^k}, \gamma_p^{n_s^k}\} \tag{12}$$

The problem as formulated above becomes an NP-hard problem. As such, solving it using conventional solvers like CPLEX or Gurobi is not feasible in terms of execution time, especially when dealing with large scale networks. Therefore, this paper proposes a heuristic approach that is able to achieve near-optimal solutions within feasible run-times.

IV. PROPOSED MULTI-STAGE GRAPH BASED DISTRIBUTED SERVICE PROVISIONING ALGORITHM

This section introduces the proposed multi-stage graph based algorithm (MuL) for multi-domain service deployments. Specifically, the steps involved in the algorithm execution, including their corresponding pseudo-codes, are described here. For the multi-domain service deployment problem, the service embedding algorithm targets to obtain a set of InPs that minimizes the service deployment cost while satisfying the request requirements. Given the large number of possible combinations for mapping the different VNFs of the request, this problem is computationally intractable. Hence, looking for exact solutions becomes unfeasible, especially for large network scenarios. This is further exacerbated by the reluctance of InPs to disclose information related to their internal topology or policies. This way, it makes conventional heuristics, such as those based on node-ranking, unfeasible for this problem. With this motivation, this paper proposes an approach that obtains the provisioning solution in three phases that will be subsequently described, with the aim to reduce the problem dimension successively. The proposal is able to obtain near-optimal solutions in practical run-times while preserving the privacy of the different InPs. The algorithm consists of 3 main steps: a Candidate InP identification, a Message exchange and a Consensus step. The proposed algorithm uses a candidate search technique to identify potential InPs that can host a fraction or the whole request. Then, these candidate InPs are used to build a multi-stage graph, where, at each stage,

all the candidate InPs of a given VNF are represented as a different node, and the interconnecting edges between the nodes of consecutive stages are the available inter-domain substrate paths. Using this multi-stage graph, a message block is constructed at the leftmost stage (source end) and propagated upstream towards the destination node. Each node, through which the message block passes, updates the received message block by increasing the cumulative mapping cost, the total cumulative delay and the list of traversed nodes, before forwarding this message block to all the nodes of the next stage. At the output end, the message block associated with the least cost value is chosen as the optimal message block, and the nodes through out which this message block was transiting are chosen as the optimal nodes/InPs for embedding the request. A detailed description of these three steps follows below:

A. CANDIDATE INPS IDENTIFICATION STEP

This step exploits that each request virtual node of a given SFC is constrained by a function/resource type and location. Similarly, the corresponding virtual links between any two virtual nodes are considered to be constrained by a delay and a bandwidth requirement. Therefore, each request virtual node of a given request may only be served by a sub-set of the available InPs that satisfy the associated constraints. The aim of this step is to associate each request virtual node with a set of InPs that can satisfy its associated constraints. Different to approaches such as in [22], in which all domains participate in the distributed solution computation, selecting a subset of InPs to participate in the solution computation minimizes the execution time of the algorithm and the amount of signaling information exchanged among the involved participants.

Whenever a request arrives, we take the orchestrator of the domain receiving that request as the Master Orchestrator (MO), and we assume that this orchestrator has access to the global information disclosed by all the InPs. This information includes the type of resources available in each domain and the boarder nodes of the different domains, which can be used to infer the span of a given domain. However, note that we are considering a restricted information disclosure. The amount of available resources, their location, the network topology or the price of each unit of resource is assumed to be private information. In this regard, the MO is responsible for comparing the specifications of the request with the global information disclosed by the orchestrators of the different administrative domains, with the goal of identifying the potential domains that could host the SFC request. The set of candidate InPs is obtained by matching the virtual nodes location and resource type constraints with the disclosed information of each InP, and also by matching the virtual links' constraints with the inter-domain links' attributes. Thus, for an InP to be among the candidate set of InPs for a given request virtual node i, the set of offered resource types disclosed by this InP must include the function type of node i, and the geographical span of that InP must include the acceptable location of this virtual node. Similarly, for two consecutive request virtual nodes i and j to be hosted by InP A and InP B, where $A \neg B$, there should exist an inter-domain path between InP A and InP B that satisfies the constraints of virtual link $i-j$. If we denote by Γ_k, $r^{n_v^p}$ and η_{vnf}^k as the geographical span of InP k, the desired location of virtual node n_v^p, and a set of VNF types that can be provisioned inside InP k respectively, then an InP is considered a potential candidate for virtual node n_v^p iff:

$$r^{n_v^p} \in \Gamma_k \tag{13}$$

$$p \in \eta_{vnf}^k \tag{14}$$

Equation 13 requires that the acceptable location of virtual node n_v^p lies within the coverage span of InP k. From equation 14, such an InP should support the resource type required by n_v^p. The pseudo-code of the candidate InPs Identification step is shown in Algorithm 1. The algorithm starts by initializing the set of candidate InPs for the request, $Cand_s^r$, to an empty set. Then, for each virtual node $n_v^p \in N_v$ of the request, the algorithm extracts all InPs that satisfy the resource type constraint, location constraint and also have a feasible connection (in terms of bandwidth and delay) between the source node S_n and destination node T_n, as potential candidates for this virtual node n_v^p, and stores these in the set $Cand^{n_v^p}$. In the case that any VNF has no potential candidate, the request is rejected at this point. Otherwise, the algorithm returns the candidate set $Cand_s^r$ made up of candidates for all the virtual nodes of the request. Note that, although the requirement to have a feasible path between candidate InPs for adjacent VNFs can be evaluated within Algorithm 1, we propose to evaluate this at the message exchange step. In this way, the number of such path computations between InPs is reduced due to the pruning of candidate nodes by virtue of location and resource type constraints.

B. MESSAGE EXCHANGE STEP

The Message exchange step can also be viewed as a distributed computation step involving each candidate InP of the request forwarding Message Blocks (MBs) to a given sub-set of candidate InPs in order to deduce a mapping solution. The Message exchange is guided by a multi-stage graph constructed by the MO and based on the obtained candidate InPs. The leftmost stage in the graph corresponds to the source node (originating InP) and the rightmost stage corresponds to the destination node (terminating InP). Each intermediate stage of the graph corresponds to a specific required VNF of the request. The nodes considered at each stage of the multi-stage graph are the candidate InPs for the provisioning of the VNF at that stage.

An example of such a multi-stage graph is shown in Fig. 4 for a request in which the traffic has to traverse three VNFs, a single source and a single destination, and candidate sets for the VNFs being: VNF1=$\{A, B\}$, VNF2=$\{A, C\}$,

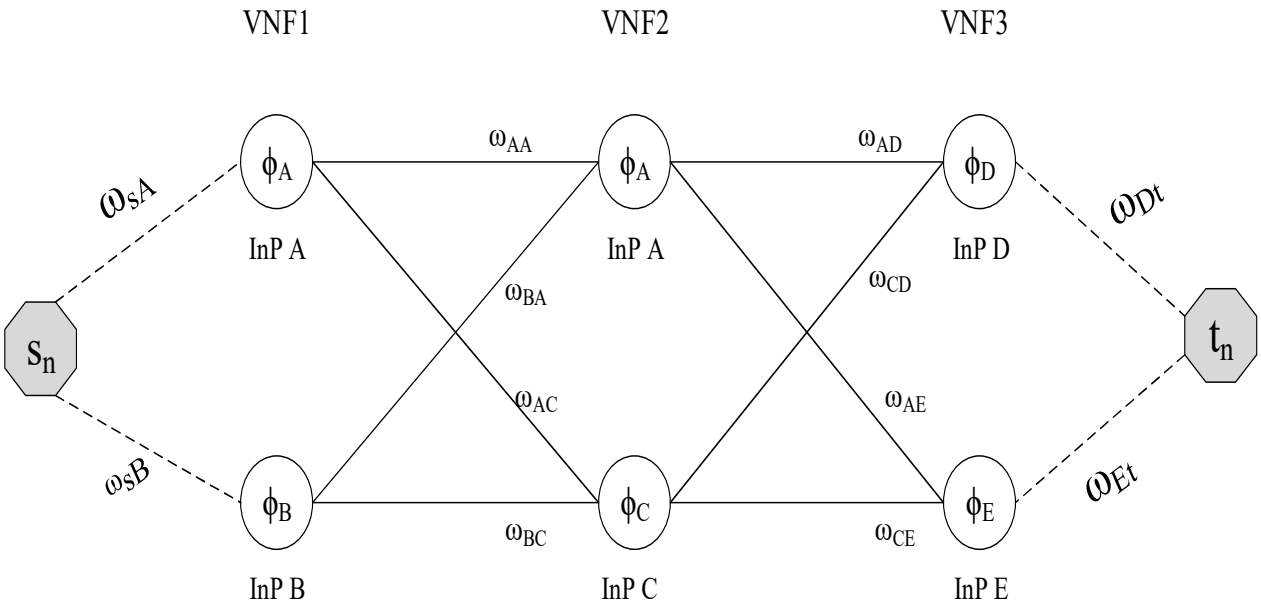


FIGURE 4: An illustration of a multi-level graph for a request whose traffic must traverse three VNFs from the ingress to egress nodes. In this case, each request virtual node has 2 InPs as potential candidates for provisioning its required VNF type, with InP A being a candidate for provisioning the VNFs for the first and second request virtual nodes

Algorithm 1: Candidate InPs Identification Algorithm

Input:G_s,G_v
Output: Candidate set,$Cand_s^r$
Initialise: $Cand_s^r$=$\emptyset$
for *Each virtual node* $n_v^p \in N_v$ **do**
  $Cand^{n_v^p} = \phi$
  for *Each Inp* $k \in \mathbb{K}$ **do**
    if $r^{n_v^p} \in \Gamma_k$ *AND* $p \in \eta_{vnf}^k$ **then**
      if *dijkstra(k,*S_n *) AND dijkstra(k,*t_n *)*$\neq Inf$ **then**
        Add k to $Cand^{n_v^p}$;
      end
    end
  end
  if $Cand^{n_v^p} = \phi$ **then**
    Reject Request
  end
  else
    Add $Cand^{n_v^p}$ to $Cand_s^r$
  end
end

VNF3=$\{D, E\}$. The connection between any two InP nodes X and Y from adjacent stages of the graph, where $X \neg Y$, corresponds to the physical path connection between the peering nodes connecting InPs X and Y. As such, the weight parameter ω_{XY} stands for the weight of that path in terms of features such as delay, residual bandwidth, and monetary cost, among others. For the particular case of the same InP being a candidate for two consecutive VNFs, (i.e. X=Y), the connection path and weight metric ω_{XY} corresponds to the intra-domain path between the candidate hosting nodes of these VNFs. Each node k in the graph is characterized by a parameter ϕ_k which represents the undisclosed information matrix of the corresponding InP. This includes the cost per unit of resource and the internal topology, among others, attributes that are only known by the specific domain orchestrator.

To understand the executed procedure of this step, and using Fig. 4 as an illustrative example of a possible multi-stage graph for an SFC with 3 VNFs, we define the following terms:

• Message Block (MB): This denotes a single message unit built as a tuple <$ID_track, Edge_track, Cost_{cum}, Del_{cum}$>. The ID_Track component, which is initialized as an empty list, stores all the IDs of the nodes/InPs that have modified the message block at the different stages (i.e. feasible candidates for the different VNFs through which the MB has traversed) from source to destination. As an example, if a message block from the source InP traverses InPs B,C,D before reaching the terminal node, then, the ID_Track for this MB at the terminal node will be , $ID_Track = [s_n, B, C, D, \tau_n]$. The $Edge_track$, initialized as an empty list, stores all the inter-domain edges that have been traversed by the message block from source to destination. Considering the above example in which $ID_Track = [s_n, B, C, D, \tau_n]$, then $Edge_track = [s_n - B, B - C, C - D, D - \tau_n$ at the terminal node. The $Cost_{cum}$ and Del_{cum} components, initialized to zero both of them, capture the cumulative cost and delay respectively along the different paths traversed by the message block (computing for both nodes and links) from source to destination. Note that each message block corresponds to a

possible mapping solution from the source node to the VNF corresponding to the last index in ID_Track. We denote by MB_n^m the message block sent from node n to node m. Note that, in this case, the stage of node n has to be to the left of that of node m.

• Message Block Set (MBS): This denotes a set of one or more message blocks.

• Optimum Message Block (MB_{opt}): This denotes the message block from the message block set that has the least cost value among all valid message blocks in that set.

• Pushing node set, k_{push}^n: The pushing node set of a given node $k \in K$ at stage n of the multi-stage graph denotes the set of all nodes in the preceding stage $(n-1)$ to which the node k has a feasible connection. Any node in such a set is called a pushing node with respect to k. As an example, the pushing node sets for the nodes in the third and fourth stages of Fig 4 are: $A_{push}^3 = \{A, B\}$, $C_{push}^3 = \{A, B\}$, $D_{push}^4 = \{A, C\}$, $E_{push}^4 = \{A, C\}$

• Receiving node set, k_{rec}^n: A receiving node set with respect to node $k \in K$ at stage n refers to the set of all nodes in stage (n+1) to which the node k has a feasible connection. As an example, the receiving node set for node C is $C_{rec}^3 = \{D, E\}$.

Then, the execution of the Message exchange step is as follows:

Starting from the leftmost stage (source node), the MO initializes N message blocks, where N is the number of candidate InPs at the next stage (i.e. the size of the receiving node set for the MO), with each message block MB_{MO}^n intended to be forwarded to a specific receiving node n. Then, for each receiving node n, it computes the shortest available path from the source node to node n, and obtains the delay, cost and the inter-domain edges constituting this path. Then, for each message block MB_{MO}^n to be forwarded to each receiving node n, the MO appends: its index into the ID_Track component, the obtained inter-domain edges into the $Edge_track$ component, and the cost and delay values to $Cost_{cum}$ and $Delay_{cum}$ components, respectively. Then, the MO forwards to each receiving node n the corresponding MB, i.e. MB_{MO}^n, for further processing. On receiving the MB, each node n at stage l ($l = 1$ if received from the source stage) identifies the receiving node set (i.e. the candidate nodes at stage $(l+1)$) from the multi-stage graph. Note that these are the candidates of the VNF to be enumerated in the next round. Then, for each node m, among the receiving candidates, node n performs the following steps:

• Obtains the optimal message block $MB_{opt}^{n,m}$ from all the message blocks it has received. The $MB_{opt}^{n,m}$ refers to the message block at node n with the least cost among the feasible message blocks to be propagated to node m. A message block is feasible to be forwarded to node m of the next stage if: i) the node m is not already part of the ID_track, unless it is the same as the current sending node (i.e. it is a candidate for both the current VNF and the next VNF, implying m=n); ii) the sum of $Delay_{cum}$ and the additional intra-domain delay to the substrate node where the VNF is to be mapped inside node n does not exceed the acceptable delay.

• Obtains the available shortest path from node n to node m. Note that this path should not include already used edges that appear in the $Edge_track$ of the $MB_{opt}^{n,m}$. This is done in order to guarantee that the user traffic from source to destination does not traverse the same edge twice. The intra-domain delay and the intra-domain cost (for nodes and links) is evaluated and added to the delay and cost of the obtained shortest path from n to m. These are then used to increase the $Delay_{cum}$ and $Cost_{cum}$, respectively, of the $MB_{opt}^{n,m}$. Also, the index of node n is added to the ID_Track component.

• Forwards $MB_{opt}^{n,m}$ to node m. Node m and the following ones will also execute the same steps until the message blocks will reach the last stage. In the case that a node is unable to push a message block to at least one of the nodes of the next stage, that node mutes all the received MBs and sends back a mute message to the MO. In the event that all the candidate nodes at a given stage have responded with a mute message, the request is rejected, and the algorithm execution stops, since this means that there is no feasible connectivity between the current VNF and the VNF corresponding to the next stage.

C. CONSENSUS AND BINDING STEP

Once the node at the last stage of the graph has computed its associated MB_{opt}, it forwards its MB_{opt} back to the MO, the MO then selects the ID_track component of the message block with the lowest cost as the definitive mapping solution for the SFC request. This is constituted by IDs of InPs that result in the least mapping solution from the source to the destination. Finally, the resources across the inter-domain links and those inside the selected domains are reserved for deploying the request.

In possible situations where the last stage could be associated with multiple nodes (e.g. in case of multiple alternative servers in which the user may access content), then each of the nodes in the last stage computes its corresponding MB_{opt} and forwards it to the MO. This, then, selects the MB_{opt} with the least cost as the definitive mapping solution. If this algorithm is to be executed in a fully distributed fashion, the execution of this step implies that the candidates of the last stage know each other (through the multi-stage graph which can be shared by the MO with all the candidate nodes). Then, once each node in the last stage of the graph has done its internal computation and evaluation, it forwards a copy of its resulting MB_{opt} to each of the other candidates in this stage. Then, each node inspects all the MB_{opt} messages at its disposal including its own. If the MB_{opt} of such a node has the lowest cost value, the node sends a "$back-off$" message to all the rest of the nodes, and it forwards its own MB_{opt} to the MO from which the definitive mapping solution is chosen as the ID_track component of the MB_{opt}, with the $Edge_track$

indicating the inter-domain edges of the solution.

Algorithm 2: Distributed Computation Step

```
Input: Multi-level Graph, G_v
Output: Mapping Solution
j=0
while j < J do
    for Each level j ∈ J do
        for Each recepient node m at level (j + 1) do
            Rec_M B^m=[]                ▷ Collect received MBs
            for Each forwarding node n at level j do
                Evaluate the optimal MB,
                 MB_opt ∈ Rec_M B^n
                Update MB_opt
                if n==Terminal node then
                    Return MB_opt       ▷ Chosen mapping
                     solution
                end
                else
                    Forward MB_opt to Rec_M B^m
                end
            end
        end
        if Rec_M B^m ∀ m at level (j+1) then
            Reject request
        end
        j=j+1
    end
end
```

For the intra-domain evaluation and mapping of the assigned sub-SFCs, an InP can use any single-substrate SFC provisioning algorithm such as [10]–[14] depending on its intra-domain policies. Moreover, under a multi-domain setting, it is possible that different InPs run different intra-domain algorithms for service provisioning. In this paper, we adopted the algorithm proposed in [10] for SFC provisioning with some minor modifications to suit the sub-SFC mapping. Using the intra-domain candidate nodes and links for the assigned sub-SFC, the algorithm constructs a multi-stage graph similar to the one in Fig. 4, with the chosen peering nodes serving as the fictitious ingress and egress nodes. The feasible nodes and links associated with the least cost between the fictitious ingress and egress nodes, considering both resource and energy consumption, are chosen for hosting the assigned sub-SFC. We refer the interested reader in the details of the algorithm to consult reference [10].

D. TIME COMPLEXITY ANALYSIS

The main steps of the proposed algorithm are: the computation of the candidate sets of InP for each VNF of the request; the processing and forwarding of message blocks from each node at each stage towards the receiving nodes of the next stage; and the selection of the InP set for the provisioning of the request. The time complexity of extracting a candidate set of InP for each request virtual node is linear in terms of the number of InPs K, $\Theta(K)$. The messages exchange step involves the use of the Dijkstra algorithm to compute the shortest path between each node i at stage n and each node j at stage (n+1) of the multi-stage graph, where $i \neq j$, as well as the evaluation of the intra-domain cost for the mapping of the VNF corresponding to stage n. The time complexity associated with the shortest path computations can be approximated as $\Theta((2C_N + (M-3)C_N^2) \times |K|log(|K|)) \approx \Theta((V-3)C_N^2) \times |K|log(|K|))$, where C_N is the number of candidate nodes for each VNF (in practice, this may be different for the different VNFs, and the same InP may be a candidate of more than one VNF). V is the number of stages in the graph, including those corresponding to the ingress and egress nodes. The time complexity of the intra-domain cost evaluation depends on the specific single domain algorithm used for the intra-domain mapping. In general, the time-complexity of the entire proposed algorithm is guaranteed to be less than $\Theta([(V-3)C_N^2) \times |K|log(|K|)] + [K \times (|N_v| - 3)N_s) \times |N_s|log(|N_s|)])$, where N_v is the number of VNFs and N_s is the number of substrate nodes for an InP. In practice, the different InPs can only support a finite number of VNFs, hence, limiting the number of candidates for each VNF. Moreover, due to the finite number of VNFs that can be supported by each InP (due to resource type and capacity constraints), the number of possible candidates for each VNF decreases as the SFC size increases, binding the time complexity of the algorithm as the SFC size increases.

V. PERFORMANCE EVALUATION

This section describes the performance evaluation of the proposed algorithm including a description of the simulated scenarios and a discussion of the obtained results. The evaluation of the proposed algorithm is made against the following bench-mark algorithms:

- Distributed Network Service Embedding (DistNSE) algorithm proposed in [5]. This work exploits the disclosed public information to compute feasible paths between source and destination, and from that the path with the least cost is chosen for mapping the service request. Considering all possible paths from source to destination to obtain all feasible solutions, the benchmark DistNSE algorithm is optimal in terms of acceptance ratio, hence, it becomes a suitable algorithm for performance benchmarking. In our comparison we considered the best performance scenario of the DistNSE algorithm, in which all feasible paths from the ingress to the egress nodes are evaluated, and from them the best path was selected.
- Multi-level Aggregation Algorithm (MuL-Ag). We designed this as a benchmark algorithm with its execution being similar to the proposed MUL. However, at each stage, instead of each node evaluating the optimal message block to be forwarded to the nodes of the next stage, that node aggregates/combines all the received messages into a message set and forwards all these to the next stage nodes as it is the case adopted by distributed algorithms in literature [22]. The target is to demonstrate the gain resulting from our proposed technique of only sending a single message from a node to another given node.

A. PERFORMANCE METRICS

We evaluate the performance of the proposed algorithm considering a number of performance metrics discussed below:

1) Average acceptance ratio, AR

This is computed as the ratio of the number of successfully accepted requests to the total number of arriving requests, i.e., the sum of both accepted and rejected. The AR metric is a direct indicator of the algorithm efficiency in using the underlying resources. Therefore, a service deployment algorithm should target a high AR performance in order to maximize the revenue returned to the Network Service Provider. This is computed as follows:

$$AR = \frac{No.\ of\ embedded\ requests}{Total\ number\ of\ requests} \tag{15}$$

2) Average provisioning cost, C(Gv)

The $C(Gv)$ is the average cost incurred by the MO on provisioning the SFCs across different domains from source to destination. This cost captures the cost of node mapping and link mapping, and it is evaluated as shown in equation 2.

3) Average revenue, Rev

This metric is used to express the average revenue, over time, obtained by the MO. This is computed as the monetary return from the use of the demanded CPU and bandwidth resources. If we denote by $rev^r(G^v)$ as the revenue received by a service provider from embedding a request $r \in R_A$, then, the total from all admitted requests is defined as:

$$Rev_{total} = \sum_{r \in R_A} rev^r(G^v) \tag{16}$$

Then, the average revenue obtained from each admitted request can be evaluated as:

$$Rev = \frac{1}{|R_A|} \sum_{r \in R_A} rev^r(G^v) \tag{17}$$

If we denote by $\gamma_c^{n_v}$ and $\gamma_{bw}^{e_v}$ as the revenue received by a SP for each unit of cpu and bandwidth resource, respectively, charged for each virtual node and virtual link of the request, the revenue obtained after embedding a given SFC request $r \in R_A$ at time $t \in T$ can be defined as below:

$$rev^r(G^v, t) = \begin{cases} \sum_{n_v \in N^v} \gamma_c^{n_v} dem_{cpu}^{n_v} + \\ \sum_{\forall e_v \in E_v} \gamma_{bw}^{e_v} dem_{bw}^{e_v} & \text{if } z_t^r\text{=1} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

where $z_t^r \in \{0, 1\}$ is a binary variable equal to 1 if resources are assigned to request $r \in R$ at time $t \in T$, zero otherwise. Therefore, the total revenue obtained in serving a request r throughout its life-time is computed as:

$$rev^r(G^v) = \sum_{t \in T} z_t^r \times rev^r(G^v, t) \tag{19}$$

where the term $\sum_{t \in T} z_t^r \leq \tau^d$ denotes the total service time of the request.

4) Average request provisioning time, Avg_T

This is the average time it takes to the service deployment algorithm to compute a mapping solution for any admitted request. Aware that future services will have stringent latency start-up requirements, a useful service deployment algorithm must work with a low Avg_T. This is computed as:

$$Avg_T = \frac{1}{|R_A|} \sum_{r \in R_A} tim_{prov}^r \tag{20}$$

where $R_A \in R$ denotes the set of all admitted requests, and tim_{prov}^r denotes the time taken by the algorithm to obtain a deployment solution for request $r \in R$.

B. SIMULATION ENVIRONMENT AND SETTINGS

Network topology: Depending on the particular experiment carried out, this work considers a substrate network composed of InPs varied from 4 to 12 participants, and with a connectivity probability between InPs fixed to 0.5. Each InP is modeled by a real network topology, namely a BIC topology as explained in [48], composed of 33 nodes and 41 edges. The resource capacity of the different intra-domain links and nodes follows a uniform distribution U(200, 300). Any intra-domain link delay, specified in milliseconds, follows a uniform distribution U(1, 6). The above settings are similar to those adopted in [38]. The cost of transmitting and processing 1 GB of data (approximately 16,384 packets of 64 KB size each) is considered to follow a uniform distribution of U($0.05, $0.12) and U($0.15, $0.22), respectively, as also adopted in [49]. This is aligned with common charging prices like those applied by Amazon EC2. The processing delay of a packet at each NF follows a uniform distribution $U(0.045ms, 0.3ms)$, with the processing delay of a service chain being the sum of the processing delays of the constituent NFs.

SFC requests: Each request $r \in R$ is generated with a random source τ_s^r and a random destination τ_d^r from G_s, with $\tau_s^r \neq \tau_d^r$, and with a packet rate demand ρ measured in packets/s following a uniform distribution $U(400, 4000)$. The end-to-end delay requirement of each request follows a uniform distribution $U(10ms, 30ms)$. We consider 5 categories of network functions: Firewalls, Proxies, NATs, DPIs and Load Balancers, with their computing resource demands adopted from [50]. The number of VNFs constituting each SFC instance is set different depending on the scenario under consideration. Considering the online case, the arrival rate of requests follows a Poisson distribution with a mean value varied from 2 to 10 requests per window of 100 time units, but this is also dependent of the experiment under consideration. Similarly, the life-time of such requests is exponentially distributed with a mean value of 1000 time units.

All simulations were carried out on a desktop computer

running a Windows Operating System with the following features: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHZ and 64GB of RAM.

C. RESULTS AND DISCUSSION

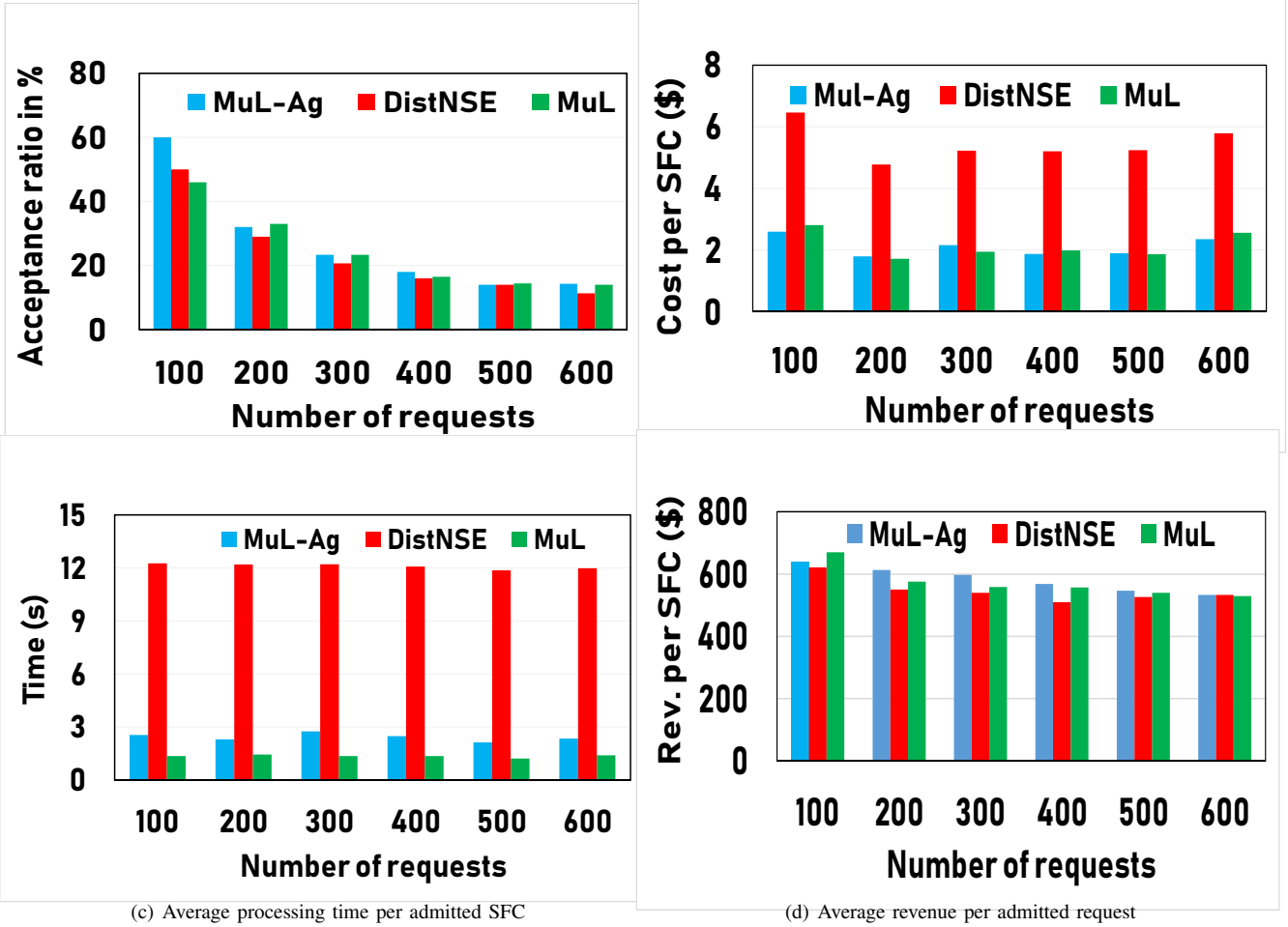
This section presents and discusses the results obtained from different online and offline experiments. Under the offline scenario, all the requests to be served, including their attributes, are known in advance. And these, once admitted, do not leave the system until the simulation ends. Therefore, the resources allocated to these requests cannot be reused by other requests. The offline scenario is useful because it gives us a clear insight into the algorithm's ability to deal with permanent loading stress conditions [2]. For the online scenario, the requests continuously arrive to the system according to a given arrival distribution, besides, any request will have a finite life-time, shorter than the simulation window. For this last scenario, the resources assigned to an accepted request are reclaimed upon the ending of the service. For each arriving request, the different steps of the algorithm are executed to provision such a request. The obtained results are discussed below.

1) Offline scenario

This section presents and discusses the results obtained from different experiments considering the offline scenario:

In Experiment 1, whose results are shown in figure 5, the impact of the demand size on the performance of the algorithms is analysed. From figure 5(a), the 3 algorithms have the same competitiveness (within a 4% margin) in terms of acceptance ratio, with an average value of: 27.0%, 23.5% and 24.5% for MuL-Ag, DistNSE and MuL, respectively, averaged across all the tested demand sizes. However, the DistNSE algorithm results in the worst performance in terms of average mapping cost per admitted SFC, with an average value of 5.45$, which is approximately a 60% higher compared to MuL and MuL-Ag, whose average cost values are: 2.11$ and 2.15$, respectively, averaged across the different demand sizes. The poor performance of the DistNSE algorithm in terms of mapping cost is attributed to the fact that in DistNSE, InPs can compete only for the previously mapped sub-SFC, as opposed to the multi-stage algorithms, in which an InP can compete for any sub-SFC of the request as long as it is a valid candidate. The results in figure 5(c) demonstrate the superior performance of MuL algorithm in terms of average processing time per admitted request, with an average value of 1.35 seconds across all demands. This translates into a performance improvement of 44.1% and 88.79% compared to MuL-Ag and DistNSE, respectively, whose average values are: 2.45 seconds and 12.1 seconds. For the MuL algorithm, each node at a given stage forwards a single message block to each receiving node at the next stage, this results in a lower processing load at the receiving nodes, hence, reducing the execution time compared to MuL-Ag, in which each node forwards all aggregated message blocks to each receiving node at the next stage. In terms of average revenue per admitted request, MuL is as competitive as MuL-Ag (within a 2% margin), and results in a 4.4% improvement compared to DistNSE, as shown in figure 5(d). Moreover, the average revenue for each admitted request tends to decrease when increasing the demand size, due to the decreased resources in the network, making it increasingly difficult to admit requests with high revenue. In summary, experiment 1 has demonstrated that MuL results in a better performance in terms of mapping cost and execution time compared to DistNSE. In terms of AR, cost and average revenue per admitted request, it is found to be as competitive as MuL-Ag, yet, achieving up to a 44.1% improvement in terms of execution time.

Experiment 2, whose results are shown in figure 6, analyses the impact of the request size, in terms of number of VNFs, on the performance of the different algorithms. Considering the results of mapping cost shown in figure 6(a), the average mapping cost per admitted request for all the 3 algorithms tends to increase as the number of VNFs per SFC increases. This is something expected since SFCs with more VNFs are associated with a higher consumption of both node and link resources, resulting in a higher provisioning cost. However, like in experiment 1, DistNSE results in the worst performance in terms of cost, with an average value of 9.97$, which is 33% higher than MuL-Ag, whose average value is 6.67$, and 44.7% higher than MuL, whose average cost value is 5.52$. The poor performance of the DistNSE in terms of mapping cost is largely attributed to the inability of the different InPs along the different paths to compete for all the sub-SFCs that they could potentially map, as they only compete for the previously mapped sub-SFC. The results of total revenue from the admitted requests is shown in figure 6(b), where the average values for DistNSE, MuL-Ag and MuL are: 10524.93$, 11486.16$, 9913.99$, respectively. This shows that the MuL is competitive in this metric with only a 5% and 13% difference compared to DistNET and MuL-Ag respectively. The results in figure 6(c) demonstrate the superior performance of the MuL compared to MuL-Alg in terms of average execution time per admitted request. In general, the average execution time per admitted SFC grows with the increase in the number of VNFs per SFC for the 3 algorithms. This is expected since each additional VNF (and hence, virtual link) comes with an extra processing time of any intra-domain mapping. However, as observed, the time complexity of MuL-Ag tends to grow exponentially when increasing the SFC size, resulting in an average value of 17.03 seconds, which is 86.3% worse than the MuL, whose value is 2.33 seconds, averaged across all SFC sizes. This is attributed to the fact that, as the number of VNFs increases, the number of stages of the multi-stage graph increases. As a result, the number of messages received by each node increases drastically for the MuL-Ag algorithm, especially for the nodes at the rightmost stages, this increases the computational load at these nodes, resulting into extremely high execution times. On the other hand, for the MuL



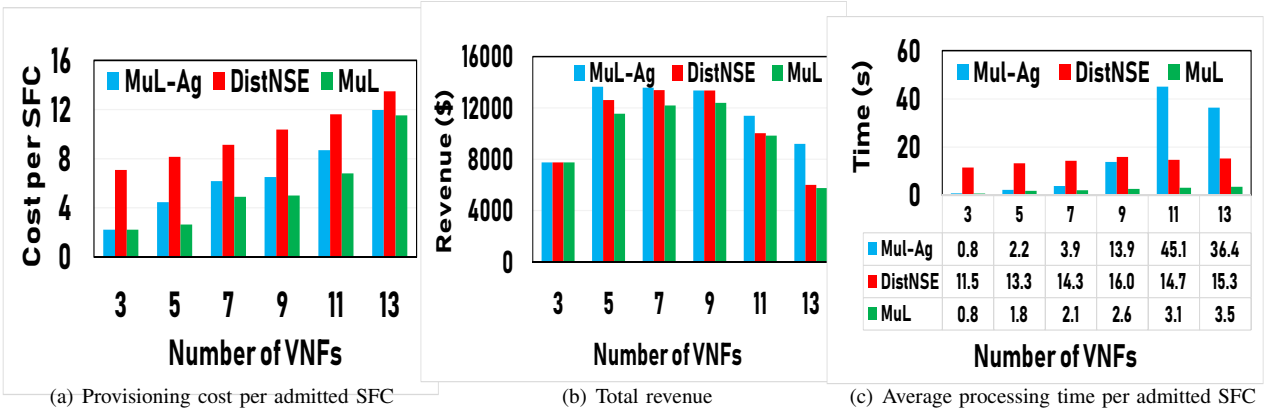
(c) Average processing time per admitted SFC

(d) Average revenue per admitted request

FIGURE 5: Experiment1: Results of experiment 1 of the offline scenario in which the number of demands is varied from 100 to 600 considering 10 InPs.

algorithm, each node forwards only a single message block to each node at the next stage. Therefore, the number of messages received by a given node at a given stage is only dependent on the number of pushing nodes in the preceding stage, and not on the stage depth of the node. The DistNSE algorithm results in a 83.4% increase in terms of execution time compared to MuL, with an average value of 14.17 seconds.

In Experiment 3 of this scenario, whose results are shown in figure 7, the impact of the substrate network size is analyzed by varying the number of InPs from 4 to 12. The 3 algorithms have a close performance in terms of AR (an approx. 4% difference) with average values of: 37.0%, 33.0% and 35.8% for the MuL-Ag, DistNSE and MuL algorithms, respectively. Moreover, the AR performance of the algorithms slightly improves as the number of InPs increases due to an increase in the amount of available resources. The results in figure 7 (b) show that the average mapping cost per admitted SFC for the 3 algorithms tends to increase as the number of InPs increases. This is attributed to the fact that, increasing the number of InPs, increases the prospects of admitting requests with more VNFs and resource requirements, which are associated with higher costs. Moreover, the probability of traversing multiple inter-domain paths between ingress and egress nodes increases as the number of InP increases. However, this figure also reveals that MuL results in a 52.5% improvement in terms of average mapping cost per admitted request compared to DistNSE, with an average value of 2.54$ compared to 5.35$ from DistNSE. The MuL-Ag results in an average value of 2.70$, representing a 5.6% difference with respect to MuL. Moreover, in terms of execution time, MuL results in a significant gain, especially as the number of InPs increases, with an average value of 0.69 seconds, averaged across the different number of InPs, as shown in figure 7(c). This translates into a performance improvement of up to 15.9% and 98.0% compared to MuL-Ag and DistNSE, respectively, whose average processing times per admitted request are: 0.83 seconds and 34.4 seconds, respectively. The exponential growth in execution time of the DistNSE algorithm results from the path computation step of the DistNSE, which requires computing all paths from source to destination, which tends to grow fast as the number of InPs increases. In a similar way, as the number of InPs increases,



(a) Provisioning cost per admitted SFC (b) Total revenue (c) Average processing time per admitted SFC

FIGURE 6: Experiment 3: Variation of number of VNFs for offline Scenario: Results of experiment 2 of the offline scenario in which the number of VNFs per SFC request is varied from 3 to 13 considering 8 InPs and demand size of 100 requests.

the number of candidate InPs (hence, nodes at each stage of the multi-stage graph) increases. This increases the number of aggregated messages that are forwarded between the different nodes of the multi-stage graph, hence, affecting the computational complexity of the MuL-Ag algorithm, since, each node forwards all feasible message blocks to its receiving nodes under this approach. The total revenue from the three algorithms is almost the same across the different substrate network sizes, as shown in figure 7(d).

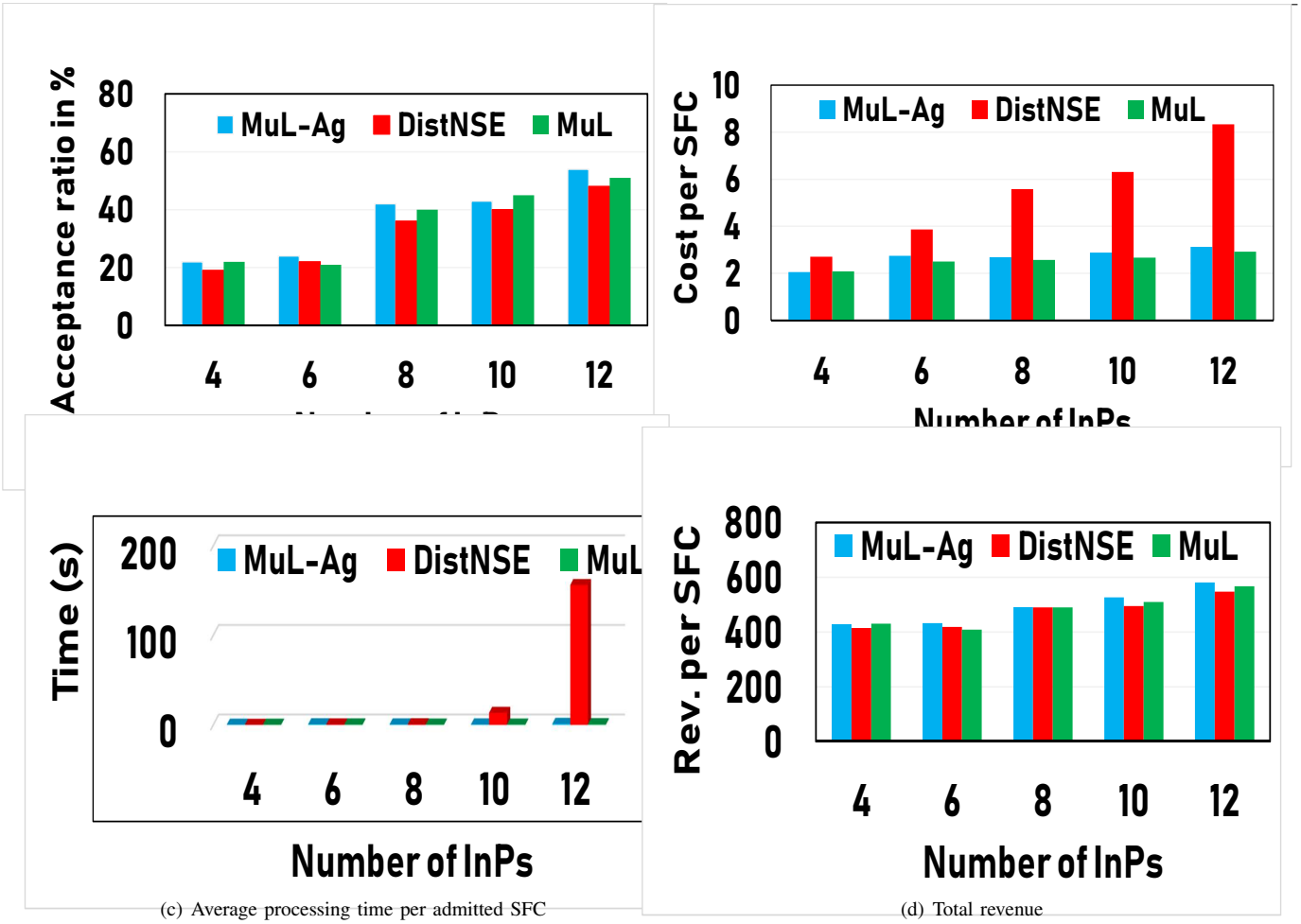
The results from the above offline experiments have demonstrated that the MuL algorithm is only 3% inferior compared to DistNSE and MuL-Ag considering the worst case scenario across all applied metrics, yet, resulting in up to 86.3% and 98.0% improvements in terms of execution time with respect to MuL-Ag and DistNSE, respectively, in some cases. Moreover, all the experiments revealed that the MuL algorithm executes in linear time. Finally, the DistNSE algorithm results in more than a 44.7% increase in terms of provisioning cost per admitted request compared to MuL for all considered experiments.

2) Online scenario

In this section we analyze the results obtained from the experiments conducted while considering online requests. The results of the different experiments for this scenario are discussed below:

Experiment 4, whose results are shown in figure 8, analyses the impact of the arrival rate of the requests. The AR performance results shown in 8(a) reveal that the AR for all algorithms decreases when increasing the arrival rate. This is expected since increasing the arrival rate results in an earlier exhaustion of the available resources, leading to an increase in the request rejection rate. Moreover, DistNSE and MuL have shown to have the same competitiveness (i.e. within less than a 1% difference) in terms of AR, with average values of: 36.03% and 37.2% for DistNSE and MuL respectively, averaged across all arrival rates. MuL-Ag results in a higher performance with an average value of 42.45% (a 5.3% improvement over MuL) , due to the fact that it forwards all possible messages, increasing chances of finding better solutions, albeit at the cost of higher run times. In terms of average cost per accepted SFC, as shown in figure 8(b), the average values of the DistNSE, MuL-Ag and MuL are: 4.80$, 3.01$ and 2.64$, respectively. Therefore, MuL results in a 44.9 % improvement in terms of mapping cost compared to DistNSE, and a 12.2% improvement compared to MuL-Ag. Moreover, all the algorithms execute in polynomial time for this scenario, with each algorithm executing even in a fraction of a second, with average values of: 42.76 milliseconds, 39.89 milliseconds and 39.67 milliseconds, for the MuL-Ag, DistNSE and MuL, respectively, as reflected in figure 8(c). The DistNSE algorithm is able to achieve this performance because this experiment uses 7 InPs, which is a relatively small number of InPs. For all the algorithms the average processing time per admitted request tends to decrease with an increase in the arrival rate. This is due to the fact that, as the arrival rate increases, the number of feasible links and nodes with enough resources decreases, resulting in a decreasing number of paths to be considered for the solution computation. From the results of 8(d), the average revenue per admitted request decreases as the arrival rate increases. This is expected since, as the rate increases, the available resources decrease, hence, the prospects of admitting requests returning a high revenue (i.e. usually those with a high number of VNFs and a high resource demand specification) decreases, hence, affecting the average revenue per admitted request. The average revenues per admitted request, averaged across the different arrival rates, for the different algorithms, are: 595.8650141$, 570.1003804$ and 577.0008115$, for MuL-Ag, DistNSE and MuL, respectively. Therefore, the MuL behaviour is inferior to MuL-Ag for less than 4%, and within a 1% margin with respect to DistNSE, in terms of revenue per admitted SFC.

In Experiment 5, whose results are shown in figure 9, the impact of the substrate network size on the algorithms' performance is analyzed. MuL and DistNSE result in similar performance in terms of average AR, with average values



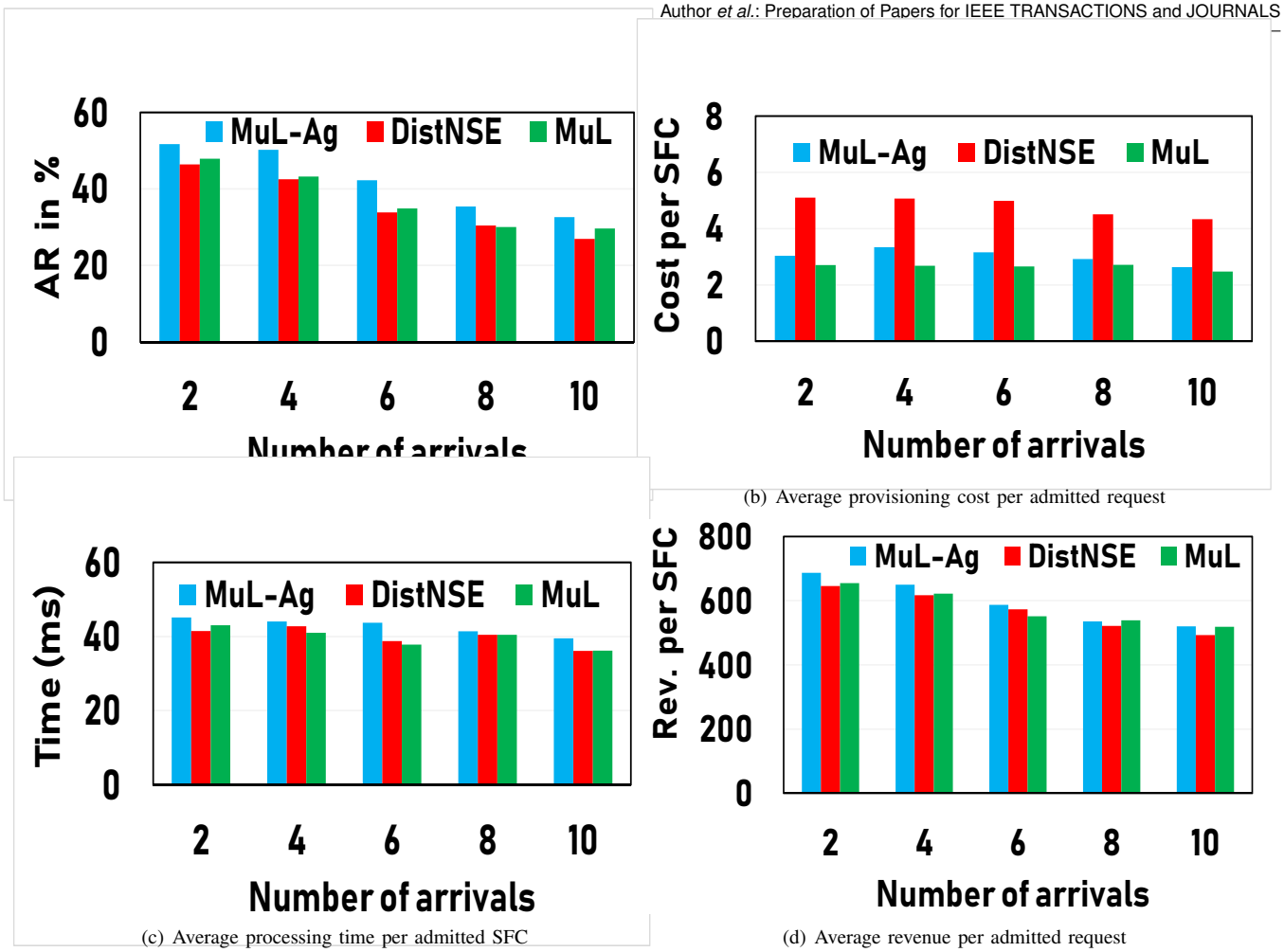
(c) Average processing time per admitted SFC

(d) Total revenue

FIGURE 7: Variation of number of InPs for offline Scenario: Results of experiment 3 of the offline scenario in which the number of InPs is varied from 3 to 12.

of: 51.25% and 51.81% respectively. MuL-Ag results in a 7% improvement in terms of AR with an average value of 58.64%. Moreover, the AR performance of all the algorithms is shown to increase when increasing the number of InPs. This is expected since increasing the number of InPs results in an increase in both node and link resources, hence, improving the AR performance. For the considered number of InPs, the average execution times in seconds per admitted request, for the three algorithms, are: 0.84, 4.54, and 0.73, for the MuL-Ag, DistNSE and MuL algorithms, respectively, averaged over all InP values as reflected in figure 9(c). This result reveals that MuL results in a 13.6% and a 83.9% improvement compared to MuL-Ag and MuL, respectively. Moreover, the execution time for all the algorithms increases when increasing the number of InPs. This is expected since this leads to an increased number of paths from source to destination for the DistNSE algorithm, and an increase in the number of nodes at each stage of the multi-stage graph of the MuL and MuL-Ag algorithms. From figure 9(b), the average mapping cost per admitted request for all the algorithms tends to increase with the number of InPs. This is attributed to the fact that, increasing the number of InPs, increases the prospects of admitting requests with more VNFs and resource requirements, which are associated with higher costs. This is evident in figure 9(d) where the average revenue per admitted request increases with increase in substrate size. In this scenario, MuL results in a 48.0% and 15,2% improvement in terms of average mapping cost compared to DistNSE and MuL-Ag, respectively: 2.98$, 4.86$ and 2.52$, for MuL-Ag, DistNSE and MuL, respectively. The results of the average revenue per accepted VNR are: 612.4$, 554.0$ and 552.7$ for MuL-Ag, DistNSE and MuL, respectively, revealing a close performance (within less than a 10% difference) among the three algorithms in terms of this metric. The average revenue per admitted request among all the algorithms increases when increasing the number of InPs. This is expected, since, with an increased availability of node and link resources, the different algorithms are able to map SFCs with a higher number of VNFs, hence, producing a greater revenue.

The results from both online and offline experiments reveal that the proposed algorithm performance is optimized in terms of acceptance ratio, execution time and embedding cost. Moreover, the simulation results further reveal that



(b) Average provisioning cost per admitted request

(c) Average processing time per admitted SFC

(d) Average revenue per admitted request

FIGURE 8: Experiment 4: Results of experiment 4 considering the online scenario with the arrival rates varied from 2 to 10 for each 100 time units for a total of 10000 time units and considering 7 InPs.

the strategy of, for each InP node in the graph, processing the received message blocks to only forward the least cost message block, significantly reduces the execution time of the algorithm without degrading its performance.

3) Computation Overhead

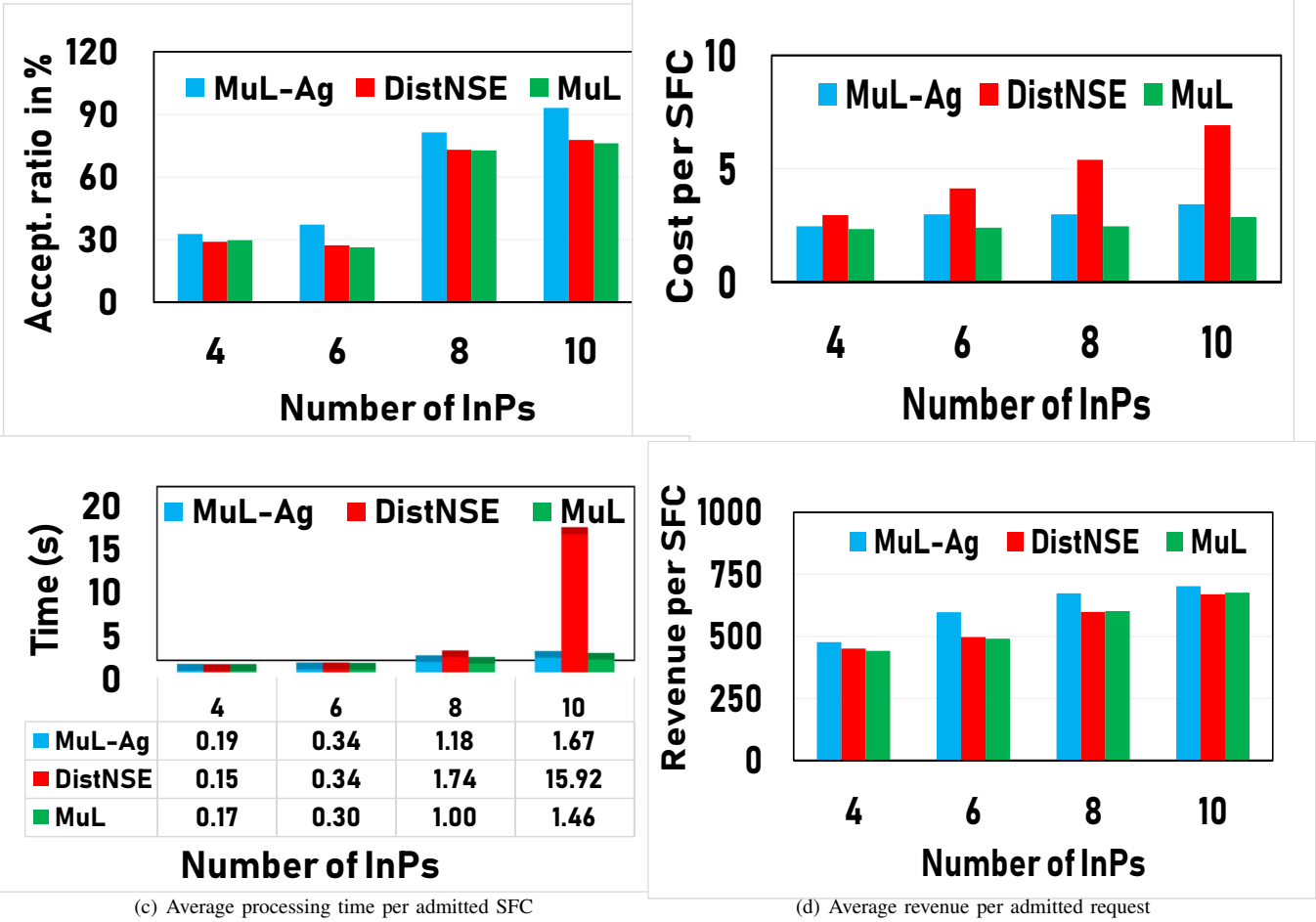
In general, distributed algorithms have an inherent drawback of high signalling overhead in terms of messages exchanged between participating nodes, especially with increasing network and request sizes. In order to evaluate the message exchange overhead involved in the proposed MuL algorithm, we denote by C_n^v as the number of candidate InPs for the VNF corresponding to stage v of the multistage graph, and denote by C_n^{v+1} as the number of candidate nodes for the stage $v+1$. Since each node in a given stage of the multi-stage graph forwards a single message to each node of the following stage of the graph, the number of messages forwarded from stage v to stage $v+1$ of the graph is evaluated as follows:

$$Msg_v^{v+1} = C_n^v \times C_n^{v+1} \tag{21}$$

In this way, the total number of messages exchanged throughout the graph is evaluated as follows:

$$Msg_{tot} = \sum_{v=1}^{v=V-1} C_n^v \times C_n^{v+1} \tag{22}$$

where $|V|$ is the total number of stages in the multi-stage graph, including those corresponding to the ingress and egress nodes. If we denote by β^v as the probability that a given InP $k \in \mathbb{K}$ is a candidate node for the VNF corresponding to stage v, then, C_n^v can be approximated as $\beta^v \times K$, where K is the total number of InPs. Therefore, from Eqn. 22, the number of messages involved in the distributed computation of the provisioning solution is increased as the number of VNFs of the request increases, since this results in an increase in the number of stages of the multi-stage graph, as shown in Fig. 10(c). Additionally, as the number of substrate nodes in the network increases, the number of possible candidates for each VNF increases, further increasing the number of messages. Therefore, by limiting the number of nodes at any stage of the graph, the total number of messages can be reduced, which is



(c) Average processing time per admitted SFC

(d) Average revenue per admitted request

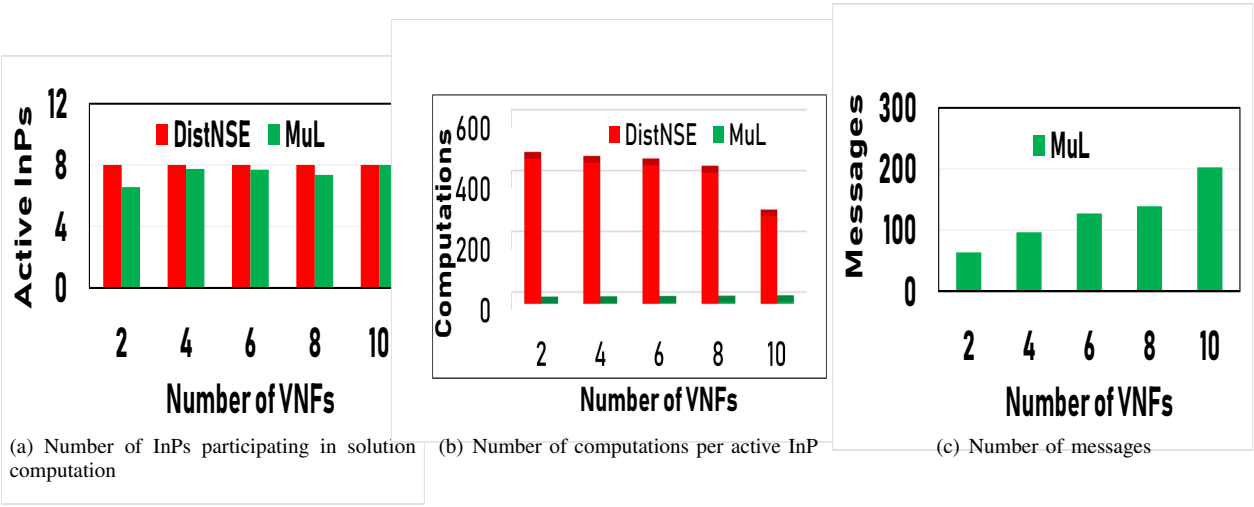
FIGURE 9: Variation of the number of InPs for online scenario: Results of experiment 2 of the online scenario with the number of InPs varied from 4 to 10 considering arrival rate of 5 requests for each 100 time units for a total of 10000 time units.

the motivation behind the candidate extraction step, which targets to consider only feasible candidates to participate in the solution computation. An elaborate evaluation of the effect of the substrate network size and SFC request size on the signalling overhead of distributed algorithms is given in [22].

In experiment 10, whose results are shown in Fig. 10, we evaluate the performance of the proposed MuL algorithm against DistNSE in terms of the number of nodes/InPs that participate in the computation of the provisioning solution for each request and the number of messages received by each node for to make a computation. From Fig. 10(a), on average, the number of InPs participating in the solution computation are 7.3 and 8 (all InPs) for MuL and DistNSE, corresponding to an 8.2% improvement of MuL over DistNSE. Moreover, from Fig. 10(b), each participating InP receives 5 and 106 messages for processing for MuL and DistNSE respectively. This performance is attributed to the fact that DistNSE relies on computing paths between ingress and egress nodes using an abstracted topology of peering nodes. In this way, it is possible for a given InP to be part of the different paths, hence participating in the computations of those paths. Moreover, even nodes that are not feasible candidates for the solution receive the sub-SFC for intra-domain provisioning evaluation during the solution computation, as long as they are part of a potential solution path.

VI. CONCLUSION

This paper has proposed a multi-stage graph based algorithm for provisioning SFCs across multiple domains while considering a limited disclosure of information from the involved InPs. The multi-stage graph is constructed from a pre-computed set of InPs obtained by a candidate search technique which enhances the run-time complexity of the algorithm thanks to reducing the set of InPs involved in the solution computation. Moreover, the simulation results have also revealed that the proposed algorithm can result in up to a 7.9 % improvement in terms of acceptance ratio, while spending a shorter execution time, in comparison with a state-of-the-art benchmark algorithm. Considering different offline and online experiments, our multi-stage algorithm is found to be scalable when increasing both the substrate network size and the request demand.



(a) Number of InPs participating in solution computation

(b) Number of computations per active InP

(c) Number of messages

FIGURE 10: Experiment 10: Analysing the message exchange performance with increase in VNF size considering 8 InPs with inter InP connectivity set to 0.3

In this work, we have considered the requests to be characterized by immutable requirements in terms of link and node resources throughout their life-time. However, in practice, such requirements may have temporal variations, requiring the embedding algorithm to intelligently adapt to such dynamism. Moreover, an elastic behaviour when considering a limited information disclosure is non-trivial, requiring the elasticity algorithm to intelligently rely on the partially disclosed information and its past experience to infer short-term future resource availability or any request requirements alteration. These last considerations will be leading our immediate future work.

REFERENCES

[1] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," Computer Networks, vol. 167, no. November, 2020.

[2] G. Kibalya, J. Serrat, J. L. Gorricho, H. Yao, and P. Zhang, "A novel dynamic programming inspired algorithm for embedding of virtual networks in future networks," Computer Networks, vol. 179, no. May, p. 107349, 2020. [Online]. Available: https://doi.org/10.1016/j.comnet.2020.107349

[3] P. Zhang, H. Yao, and Y. Liu, "Virtual Network Embedding Based on Computing, Network, and Storage Resource Constraints," IEEE Internet of Things Journal, vol. 5, no. 5, pp. 3298–3304, 2018.

[4] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding," IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 98–112, 2019.

[5] A. Abujoda and P. Papadimitriou, "DistNSE: Distributed network service embedding across multiple providers," 2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016, no. i, pp. 1–8, 2016.

[6] G. Kibalya, J. Serrat, J. L. Gorricho, R. Pasquini, H. Yao, and P. Zhang, "A Reinforcement Learning Based Approach for 5G Network Slicing across Multiple Domains," in 15th International Conference on Network and Service Management, CNSM 2019, 2019.

[7] P. Kaliyammal Thiruvasagam, V. J. Kotagi, and C. S. R. Murthy, "The More the Merrier: Enhancing Reliability of 5G Communication Services With Guaranteed Delay," IEEE Networking Letters, vol. 1, no. 2, pp. 52–55, 2019.

[8] P. Zhang, C. Wang, Z. Qin, and H. Cao, "A multidomain virtual network embedding algorithm based on multiobjective optimization for Internet of Drones architecture in Industry 4.0," Software - Practice and Experience, no. October 2019, pp. 1–19, 2020.

[9] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A Resource Allocation Framework for Network Slicing," Proceedings - IEEE INFOCOM, vol. 2018-April, pp. 2177–2185, 2018.

[10] G. Kibalya, J. Serrat, J. L. Gorricho, J. Serugunda, and P. Zhang, "A multi-stage graph based algorithm for survivable Service Function Chain orchestration with backup resource sharing," Computer Communications, vol. 174, no. May 2020, pp. 42–60, 2021. [Online]. Available: https://doi.org/10.1016/j.comcom.2021.04.008

[11] X. Shang, Z. Li, and Y. Yang, "Placement of highly available virtual network functions through local rerouting," Proceedings - 15th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2018, pp. 80–88, 2018.

[12] ——, "Rerouting Strategies for Highly Available Virtual Network Functions," IEEE Transactions on Cloud Computing, vol. PP, no. c, p. 1, 2019.

[13] K. Karra and K. M. Sivalingam, "Providing Resiliency for Service Function Chaining in NFV systems using a SDN-based approach," 2018 24th National Conference on Communications, NCC 2018, pp. 1–6, 2019.

[14] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A link failure recovery algorithm for Virtual Network Function chaining," Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management, pp. 213–221, 2017.

[15] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay," IEEE Communications Magazine, vol. 56, no. 2, pp. 44–51, 2018.

[16] G. S. Aujla, N. Kumar, A. Y. Zomaya, and R. Ranjan, "Optimal decision making for big data processing at edge-cloud environment: An SDN perspective," IEEE Transactions on Industrial Informatics, vol. 14, no. 2, pp. 778–789, 2018.

[17] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-Provider Service Chain Embedding With Nestor," IEEE Transactions on Network and Service Management, vol. 14, no. 1, pp. 91–105, 2017.

[18] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-Provider Virtual Network Embedding With Limited Information Disclosure," IEEE Transactions on Network and Service Management, vol. 12, no. 2, pp. 188–201, 2015.

[19] F. Samuel, M. Chowdhury, and R. Boutaba, "PolyViNE: Policy-based virtual network embedding across multiple domains," Journal of Internet Services and Applications, vol. 4, no. 1, pp. 1–23, 2013.

[20] A. Song, S. Member, W.-n. Chen, S. Member, T. Gu, H. Yuan, S. Kwong, and J. Zhang, "With Historical Archives and Set-Based Particle Swarm Optimization," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 1, pp. 927–942, 2021.

[21] H. Cao, Y. Zhu, L. Yang, and G. Zheng, "A Efficient Mapping Algorithm with Novel Node-Ranking Approach for Embedding Virtual Networks," IEEE Access, vol. 5, pp. 22 054–22 066, 2017.

[22] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Vertex-centric computation of service function chains in multi-domain networks," IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conference and Workshops:

Software-Defined Infrastructure for Networks, Clouds, IoT and Services, pp. 211–218, 2016.
[23] E. G. N. . V1.1.1, "NFV-Use Cases," IEEE Network, vol. 1, no. 5, pp. 1–50, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4626228
[24] G. Report, "Network Functions Virtualisation (NFV) Release 3 ;," vol. 1, pp. 1–59, 2018.
[25] T. Mahmoodi, H. Van Helvoort, and S. Mansfield, "Management and orchestration," IEEE Communications Standards Magazine, vol. 1, no. 4, p. 60, 2017.
[26] V. Eramo, F. G. Lavacca, T. Catena, M. Polverini, and A. Cianfrani, "Effectiveness of Segment Routing Technology in Reducing the Bandwidth and Cloud Resources Provisioning Times in Network Function Virtualization Architectures ," no. i, pp. 1–20, 2019.
[27] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-domain virtual network embedding with limited information disclosure," 2013 IFIP Networking Conference, IFIP Networking 2013, vol. 12, no. 2, pp. 188–201, 2013.
[28] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang, and A. V. Vasilakos, "Low-latency and Resource-efficient Service Function Chaining Orchestration in Network Function Virtualization," IEEE Internet of Things Journal, vol. PP, no. c, p. 1, 2019.
[29] P. Zhang, H. Yao, C. Qiu, and Y. Liu, "Virtual network embedding using node multiple metrics based on simplified electre method," IEEE Access, vol. 6, pp. 37 314–37 327, 2018.
[30] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, "Security-Aware Virtual Network Embedding Algorithm based on Reinforcement Learning," IEEE Transactions on Network Science and Engineering, vol. 4697, no. c, pp. 1–11, 2020.
[31] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," IEEE Access, vol. 4, pp. 8572–8580, 2016.
[32] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," Computer Networks, vol. 55, no. 4, pp. 1011–1023, 2011.
[33] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," IEEE International Workshop on Quality of Service, IWQoS, pp. 61–70, 2014.
[34] S. Li, M. Y. Saidi, and K. Chen, "Multi-domainvirtualnetwork embedding with coordinated link mapping," Advances in Science, Technology and Engineering Systems, vol. 2, no. 3, pp. 545–552, 2017.
[35] J. Martin-Perez and C. J. Bernardos, "Multi-Domain VNF Mapping Algorithms," IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB, vol. 2018-June, 2018.
[36] Q. Xu, D. Gao, H. Zhou, W. Quan, and W. Shi, "An energy-aware method for multi-domain service function chaining," Journal of Internet Technology, vol. 19, pp. 1727–1739, 2018.
[37] K. D. Joshi and K. Kataoka, "pSMART: A lightweight, privacy-aware service function chain orchestration in multi-domain NFV/SDN," Computer Networks, vol. 178, no. May, p. 107295, 2020. [Online]. Available: https://doi.org/10.1016/j.comnet.2020.107295
[38] 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, 2018.
[39] G. Sun, Y. Li, G. Zhu, D. Liao, and V. Chang, "Energy-efficient service function chain provisioning in multi-domain networks," IoTBDS 2018 - Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, vol. 2018-March, no. January, pp. 144–163, 2018.
[40] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," Future Generation Computer Systems, vol. 91, pp. 347–360, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.09.037
[41] L. Gupta, R. Jain, A. Erbad, and D. Bhamare, "The P-ART framework for placement of virtual network services in a multi-cloud environment," Computer Communications, vol. 139, no. January, pp. 103–122, 2019. [Online]. Available: https://doi.org/10.1016/j.comcom.2019.03.003
[42] D. Bhamare, M. Samaka, and A. Erbad, "Exploring microservices for enhancing internet QoS," no. April, pp. 1–18, 2018.
[43] P. Cappanera, F. Paganelli, and F. Paradiso, "VNF placement for service chaining in a distributed cloud environment with multiple stakeholders," Computer Communications, vol. 133, no. October 2018, pp. 24–40, 2019. [Online]. Available: https://doi.org/10.1016/j.comcom.2018.10.008
[44] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," IEEE International Conference on Communications, pp. 5634–5640, 2008.
[45] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, O. Carlos, and M. Bandeira, "Orchestrating Virtualized Network Functions," vol. 13, no. 4, pp. 725–739, 2016.
[46] N. Tastevin, M. Obadia, and M. Bouet, "A Graph Approach to Placement of Service Functions Chains," pp. 134–141, 2017.
[47] K. Hejja and X. Hesselbach, "Online power aware coordinated virtual network embedding with 5G delay constraint," Journal of Network and Computer Applications, vol. 124, no. February, pp. 121–136, 2018. [Online]. Available: https://doi.org/10.1016/j.jnca.2018.10.005
[48] "The Internet Topology Zoo." [Online]. Available: http://www.topology-zoo.org/dataset.html
[49] G. Yuan, Z. Xu, B. Yang, W. Liang, W. Koong, D. Tuncer, A. Galis, G. Pavlou, and G. Wu, "Fault tolerant placement of stateful VNFs and dynamic fault recovery in cloud networks," Computer Networks, vol. 166, p. 106953, 2020. [Online]. Available: https://doi.org/10.1016/j.comnet.2019.106953
[50] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, F. Huici, and I. Nsdi, "ClickOS and the Art of Network Function Virtualization This paper is included in the Proceedings of the," 2014.

GODFREY KIBALYA received a BSc. degree in Telecommunications Engineering in 2010 from Makerere University Uganda and a MSc. degree in Telecommunications Engineering from the University of Trento, Italy. Currently, he is a Ph.D. candidate at the Technical University of Catalonia (UPC), Spain under the Department of Network Engineering. His research interests include Network function virtualization, and application of Artificial Intelligence in network management.

JOAN SERRAT-FERNANDEZ received the degree of Telecommunication Engineer in 1977, and the Doctor degree in Telecommunication Engineering in 1983, both from Universitat Politècnica de Catalunya -UPC-. Currently he is Full Professor at UPC where he has been involved in several collaborative projects with different European research groups, both through bilateral agreements or through participation in European funded projects. His topics of interest are in the field of autonomic networking and service and network management.

JUAN-LUIS GORRICHO received a Telecommunication Engineering degree in 1993, and a Ph.D. degree in 1998, both of them from the Technical University of Catalonia (UPC). Since 1994 he joined the Department of Network Engineering at the UPC, as associate professor since 2001. His most recent research interests have been focused on the management of resources for virtualized networks and functions, cloud computing and software defined networks.

DOREEN GIFT BUJJINGO received the B.Sc. degree in Telecommunications Engineering from Makerere University in 2016.She is currently a masters student in Telecommunications Engineering at Makerere University. Her research interests include Software Defined Networks (SDNs), Virtualized network functions, Resource management and allocation using machine learning.

JONATHAN SERUGUNDA received the B.Sc. degree in Electrical Engineering from Makerere University in 2005. He received the M.Sc. degree in Communication Engineering from the University of Manchester, United Kingdom in 2008 and the Ph.D. degree in Electrical and Electronic Engineering from the University of Bristol, United Kingdom in 2015. He is a lecturer at Makerere University in the Department of Electrical and Computer Engineering. He is a member of netLabs!UG which is a wireless research center based at Makerere University. His research interests include radio wave propagation and antenna design, design and analysis of wireless networks, physical layer security and unmanned aerial vehicles (UAVs) assisted communication systems.