Towards security-aware 5G slice embedding

Houda Jmila, Gregory Blanc

Version of Record: https://www.sciencedirect.com/science/article/pii/S0167404820303485
Manuscript_2611af9012a6beeffc291f92368dac2e

Towards Security-Aware 5G Slice Embedding

Houda Jmila, Gregory Blanc

SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris
9 rue Charles Fourier, 91011 Evry-Courcouronnes Cedex, France

Abstract

5G networks tend to be multi-layered, multi-party and multi-tenant to satisfy the increasing and varying user demand. Network slicing is an efficient solution to share the physical network between different and independent virtual networks called *slices*. While many research works focused on the slice resource allocation, called *slice embedding*, little attention was paid to the security aspects. In this paper, we deal with *security-aware 5G slice embedding*. This consists in satisfying the slice request while meeting its security needs. In a first step, we model the common security needs and best practices for 5G slices. This is used to design *a security-aware slice request*. Second, we devise a slice embedding algorithm that respects the security constraints. Third, we analyse the scalability of making slice embedding security-aware, in terms of acceptance rate, cost to revenue ratio, execution time and physical resource utilization.

Keywords: 5G, slice embedding, resource allocation, security, VNE.

Nomenclature

CN	Core Network
ILP	Integer Linear Programming
NFV	Network Function Virtualization
RAN	Radio Access Network
SA-CNSE	Security-Aware Core Network Slice Embedding
SDN	Software-Defined Networking
SFC	Service Function Chaining
VNF	Virtual Network Function
VNSF	Virtual Network Security Function

1. Introduction

The 5G mobile networks are expected to support a wide range of application scenarios according to a report from Huawei, HKT and GSA (2019), including autonomous driving, virtual reality, wireless healthcare and smart agriculture. These scenarios, also called *verticals* have very different requirements and expectations. The "one-network-fits-all" architecture of traditional mobile networks, where service customization is given little consideration, is no longer suitable. Recently, the *network slicing* concept (cf. Fig. 1) was introduced to support the verticals diversity. It enables the creation, for each application scenario, of a tailored virtual network, called *network slice* that satisfies its specific needs. A network slice is an end-to-end (i.e., from radio access network (RAN) to core network (CN)) logical network composed of virtualized and non-virtualized resources running on a common underlying infrastructure.

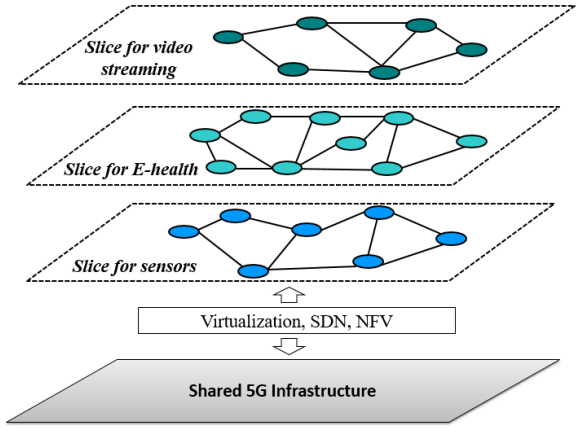


Figure 1: The Network Slicing concept

The 5G network slicing can mainly be achieved thanks to two key technologies: Software-Defined Networks (SDN) and Network Function Virtualization (NFV). SDN decouples network control from data forwarding to enable network programmability and centralized control Kreutz et al. (2015). NFV decouples network functions such as load balancers, WAN optimizers, or Intrusion Detection Systems (IDSs) from dedicated hardware and run them as software, called Virtual Network Functions (VNFs), in a cloud computing infrastructure Mijumbi et al. (2016). Thanks to SDN and NFV, network slices can be created and customized in a dynamic and flexible manner Ordonez-Lucena et al. (2017), thus reducing capital and operational expenditures (CAPEX & OPEX).

One main issue related to the 5G network is the resource

management problem. More particularly, the *slice embedding* problem, also called *network slicing*, is challenging. It consists of enabling different heterogeneous slices to share a limited pool of resources in a transparent and isolated manner. There are two types of network slicing Su et al. (2019): RAN slicing and CN slicing . The RAN slicing concerns the sharing and management of *spectrum resources* whereas the CN slicing aims to distribute the *infrastructure computing and networking resources* among multiple slices. In this paper, we focus on the CN slicing. Although this problem was extensively investigated in the literature Bhamare et al. (2016); Herrera & Botero (2016); Xie et al. (2016), little attention was paid to the security aspects. In fact, many works ignore the security needs of running services. For instance, attacks targeting the service from another slice, attacks targeting the infrastructure hosting the service, isolation between the services, among others, should be considered when deploying the slice.

To fill this gap, we focus on the **S**ecurity-**A**ware 5G **C**ore **N**etwork **S**lice **E**mbedding, referred to as the **SA-CNSE** problem in this paper. It consists in embedding the slice request while considering its security requirements. We follow a security by design approach and tackle the problem in two stages. First, we design a "security-aware" slice request to describe the security needs. Second, we embed the slice in a security-aware manner. Furthermore, we analyse how complying with security constraints impacts the slice embedding in terms of acceptance rate, embedding cost and execution time.

This work extends and completes our previous works Jmila & Blanc (2019); Boutigny et al. (2020) which had the following contributions:

i) In Jmila & Blanc (2019), we studied the resources allocation problem in the NFV environment, more particularly the Service Function chaining problem Bhamare et al. (2016).

- We showed how a basic service function chain (SFC) (composed of chained VNFs) representing a service request should be enriched to take into consideration the service security needs.
- We illustrated our proposal through a use case to demonstrate its feasibility.

ii) In Boutigny et al. (2020), we investigated the security-aware 5G slice embedding problem in a multi-domain environment, i.e. when the 5G network is supported by different infrastructure providers.

- We proposed a constraint generation model to translate security requirements into mathematical constraints.
- The solution was validated through a prototype running on a use case.

This paper extends and enhances our previous work as follows:

i) We focus on the 5G slice embedding problem. Unlike Boutigny et al. (2020), we focus on a single domain scenario. We design a security-aware 5G slice request by adapting the algorithm of Jmila & Blanc (2019).

ii) We investigate the *security-aware slice embedding* issue. We propose an Integer Linear Programming (ILP) formalization of the **SA-CNSE** problem. We conceive a heuristic solution and show how a generic embedding algorithm can be enhanced to take into consideration the security requirements.

iii) We analyse in depth the impact of handling security constraints on the slice embedding process, in different scenarios, regarding various metrics. To the best of our knowledge, this is the first research paper providing such contribution. We also compare security-aware to non security-aware embedding solutions. Thus, unlike our previous work (Boutigny et al. (2020); Jmila & Blanc (2019)) where the proposed models were assessed in specific use cases, our model is evaluated in different scenarios.

iv) We have developed a C++ simulator to perform extensive simulations. We implemented and evaluated the security-aware embedding algorithms.

v) We provide a clear and structured state of the art of the CN slice embedding problem including both legacy and security-aware approaches.

The remainder of this paper is structured as follows: Section 2 introduces the CN slice embedding problem and describes the related work. Section 3 details the security-aware slice request design. Section 4 describes the security-aware slice embedding algorithm. Section 5 evaluates the security-aware slice embedding. Section 6 concludes the paper.

2. State of the art

In this section, we first introduce the CN slice embedding problem (CNSE) and then present the related work.

2.1. The 5G CN slice embedding problem

5G CN slicing is a promising technology to establish customized end-to-end logic networks including dedicated and shared resources, in particular when dealing with multiple domains and heterogeneous technologies. The application of CN slicing introduces the problem of efficient resource management. The physical resources, used to host the slice components, have a finite amount of compute, memory and storage capacity. Physical links connecting these resources have also limited amount of bandwidth. Therefore, efficient resource management is required to yield the economical benefits promised by the 5G networks.

A slice request can be modeled by a virtual network request composed of virtual nodes representing the VNFs, and virtual links connecting them to steer the traffic. The slice embedding problem consists in mapping the VNFs and links composing

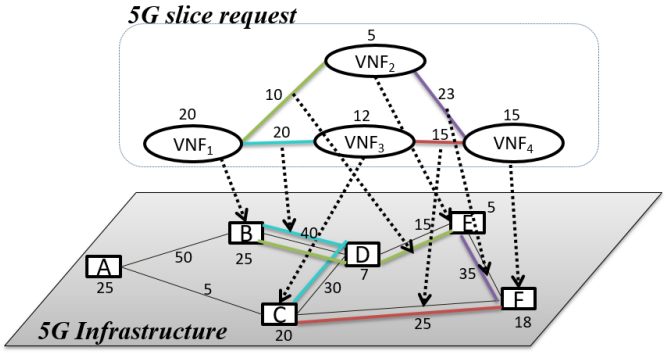


Figure 2: The slice embedding problem

the slice request to the substrate nodes and paths constituting the physical infrastructure. An efficient mapping should satisfy the slice resource needs while using fewer physical resources.

Fig. 2 (upper part) shows an example of a 5G slice request composed of four VNFs (denoted by VNF_1 to VNF_4) connected by virtual links. The numbers on top of the VNFs represent the amount of required resources (typically memory, CPU and storage) and the numbers over the links represent the required bandwidth. Similarly, in the lower part of Fig. 2, a substrate graph, representing the 5G infrastructure and the available resources is depicted (the numbers below the nodes and below/over the links represent the amount of available resources). A slice embedding scheme is possible when each VNF and virtual link of the slice request is mapped to a physical node or link having enough available resources. A mapping example is illustrated by the dotted arrows showing the mapping of each virtual element to a physical host, namely, the VNF_1 requiring 20 units of resources is mapped to the substrate node *B which has enough resources* (25 units); similarly VNF_2 is mapped to substrate node E, VNF_3 to C and VNF_4 to F. Likewise, each virtual link is mapped to a substrate path having enough bandwidth to host it. It is important to note that a path may be composed of more than one physical link: for instance, the virtual link between VNF_1 and VNF_3 is mapped onto a path composed of the physical links $B-D$ and $D-C$.

2.2. Related work

In this section, we provide a state of the art of the CN slice embedding problem. In addition, we analyse the literature of a very close problem called the *Virtual Network Embedding problem* (VNE) Fischer et al. (2013). VNE is an old and well investigated issue in the Network virtualization field Chowdhury & Boutaba (2010). It deals with sharing a common Substrate Network (SN) among different Virtual Networks (VNs). As mentioned in many research papers Li et al. (2019a); Su et al. (2019); Boutigny et al. (2020), the CN slice embedding problem can be converted to the VNE problem: the virtual networks represent the slices and the substrate network represents the 5G infrastructure. We believe that analysing the VNE literature can inspire innovative CN slice embedding approaches.

2.2.1. Traditional resource allocation algorithms

Solving the CN slice embedding and the VNE problems is NP-hard since they are related to the multi-way separator problem Fischer et al. (2013): even when all the VNFs are mapped, mapping each virtual link to a single substrate path is an unsplittable flow problem, which is also NP-hard. Therefore, most solutions are based on heuristics Fischer et al. (2013); Bhamare et al. (2016). By varying the constraints and objectives, the problem can be tackled in different manners. We briefly present the main categories found in the literature.

- **Mapping coordination.** The slice embedding problem can be decomposed into two sub-problems: the VNF mapping problem and the link mapping problem. *Two-stage slice embedding algorithms* consist in solving each sub-problem in an isolated and independent way. Generally, the VNF mapping is performed first to provide the input for the link mapping phase. Typical approaches apply a greedy algorithm that maps the VNFs requiring larger resources to the substrate nodes providing larger resources. Next, virtual links are mapped to the substrate paths using the shortest or k-shortest path Cao et al. (2017); Zhang et al. (2018); Yuan et al. (2019). In the *one-stage slice embedding approach*, virtual links are mapped at the same time as VNFs. When a VNF pair is mapped, the virtual link between them is also mapped, and so are the virtual links connecting it with already embedded VNFs Cao et al. (2019a, 2020); Li et al. (2019b).

- **Multi-domain vs. single-domain.** A CN slice request can be satisfied by a single or multiple 5G infrastructure providers. In a *single-domain scenario*, the request is fulfilled by only one infrastructure provider. Differently, in the *multi-domain scenario* Li et al. (2016, 2017); Ni et al. (2019), a coordinator (the service provider) splits the request into several sub-requests distributed to several infrastructure providers. The provided sub-networks are then interconnected with external links to form the slice.

- **Centralized vs. distributed.** In *centralized* systems, like those proposed by Soualah et al. (2019); Shahriar et al. (2019), a global view of the infrastructure state is required to take decisions according to the up-to-date description of the available substrate resources . To address scalability issues in large networks and reduce communication cost and synchronization overhead, some *distributed* slice embedding solutions (like Esposito et al. (2013, 2016); Song et al. (2019)) were proposed.

In addition to these legacy approaches, the use of **Reinforcement Learning** (RL) techniques Sutton & Barto (1998) is more and more investigated. For example, He et al. (2018) propose a multi-objective VNE embedding algorithm based on Q-learning to optimize conflicting objectives, namely energy saving and requests acceptance rate. Dolati et al. (2019) design DeepViNE, a deep RL-based VNE. Both the VN request and substrate network are encoded as two-dimensional images, which are then perceivable by a convolutional deep neural network (CNN) Krizhevsky et al. (2012) to generate the mapping

scheme. Kibalya et al. (2019) address the multi-domain slice embedding problem and propose an RL-based algorithm for partitioning the slice request to the different infrastructure providers. Sciancalepore et al. (2019) propose an RL-based broker to i) efficiently forecast future traffic levels per network slice, ii) optimize slice admission control using traffic prediction and iii) perform slice traffic scheduling.

2.2.2. *Security-aware approaches*

Few approaches examine the security facet of the slice embedding problem. Most research work focus on adding a chain of virtual network security functions to the 5G slice topology to perform some security operations. Research works concentrate on efficiently placing the VNSFs in the 5G infrastructure. The VNF placement problem is commonly known in the literature as the *Service Function Chaining* (SFC) problem Bhamare et al. (2016). Doriguzzi-Corin et al. (2017, 2019) propose an ILP formulation for placing VNSFs with respect to both *the quality of service requirements* and *the security constraints*. In Guan et al. (2018), the authors focus on searching an optimal placement for VNSFs. The best hosts are the most capable to control the traffic. This is measured by the node centrality that represents the degree of connectivity between nodes. Liu et al. (2017) tackle the problem of efficient *Security Service Chain (SSC)* deployment. An SSC is an ordered set of security functions composing a logical security service. The authors proposed some heuristic algorithms to select hosting nodes and establish routing paths.

Some security-aware VNE approaches define security constraints for VN requests and then address the security-aware mapping. In Bays et al. (2012); Bays et al. (2014), the authors consider end-to-end communication confidentiality and map the end nodes of virtual links to physical routers enabling data encryption and decryption. Similarly, Xing et al. (2013) consider embedding virtual resources in *trusted* physical resources. They define a *security protection level* to measure the ability of a physical resource to protect its hosts. In Xing et al. (2013); Li et al. (2019a); Zhang et al. (2020), the authors use the concept of *Security Level* to assist the VNE mapping decision. A virtual resource requiring a certain level of security can only be mapped to a physical resource having higher or equivalent security level. However, the security level concept remains abstract and difficult to determine in reality Boutigny et al. (2020). In Shameli-Sendi et al. (2017), the authors argue that virtual nodes can be placed in the same host to avoid security breaches between sensitive virtual nodes. In Boutigny et al. (2020), the authors support exclusion security constraints to avoid sharing physical resources with other tenants for isolation purposes.

In this paper, inspired by both the VNE and CNSE literature, we judiciously define security constraints for 5G slices and conceive a security-aware resource allocation scheme. Moreover, unlike all existing works, we deeply analyse the scalability of making 5G CN slice embedding security-aware, in terms of acceptance rate, cost to revenue ratio and execution time, through extensive simulation study. Our proposal is detailed in the next sections.

3. Securing the slice request design

The need to secure the slice embedding process is justified by the large threat landscape faced by 5G networks (cf. Khan et al. (2020); Arfaoui et al. (2018)) and the network slicing technology in particular (cf. Olimid & Nencioni (2020)). In the next section, we illustrate some possible threat targets that can be protected using our solution. Later on, we motivate the need to design a security-aware slice request.

3.1. *Threat model for 5G slice embedding*

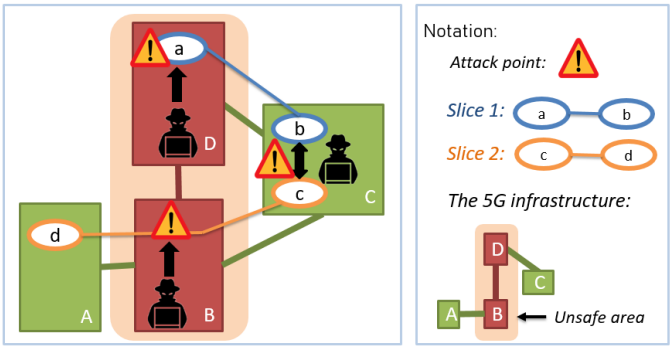


Figure 3: Representative attack points for 5G slicing

Figure 3 shows two slice requests, *Slice 1* and *Slice 2*. *Slice 1* in composed of the interconnected VNFs *a* and *b*. Similarly, *Slice 2* is composed of the VNFs *c* and *d*. These slices are hosted by a 5G infrastructure which is composed of four connected substrate nodes (*A*,*B*,*C*,*D*).

The 5G network is designed to support the multi-provider scenario where the infrastructure is provided by different providers. This introduces new security vulnerabilities as the infrastructure will be partitioned into *domains of different security levels* depending on the trust towards each provider. The colors red and green in the infrastructure topology illustrate two domains of different security levels. Namely, the red components (nodes *B* and *D* and the link connecting them) constitute an unsafe area, while the green components constitute a safe one. Below, we describe some possible attacks:

Unauthorized node access. If the VNF *a* is hosted on an insecure physical node *D*, it could be physically attacked, which can damage the quality of the service.

Unauthorized link access. If the traffic between VNFs *c* and *d* is routed over an insecure physical path (the one traversing *B*), replay and man-in-the-middle attacks could be performed.

Co-location attack. Hosting the VNFs *b* and *c* of different slices (*Slice 1* and *Slice 2*, respectively) on the same physical host, *C*, may threaten the isolation between the services and lead to security breaches (sensitive data transmitted between the slices). The same goes for virtual links sharing the same physical path.

Unauthorized access to the services. The unauthorized access to the services running over the slices may introduce privacy concerns and can impact the consumption of resources, leading to potential DoS attacks.

3.2. Motivation for designing a security-aware slice request

Considering, from the beginning, the security needs of a service running over the slice leads to determine better embedding solutions as it enables a richer and more complete description of the slice. Otherwise, addressing the slice security constraints as *an independent second step*, after the embedding, may lead to sub-optimal resource allocation. In the next subsection, we illustrate this idea through an example.

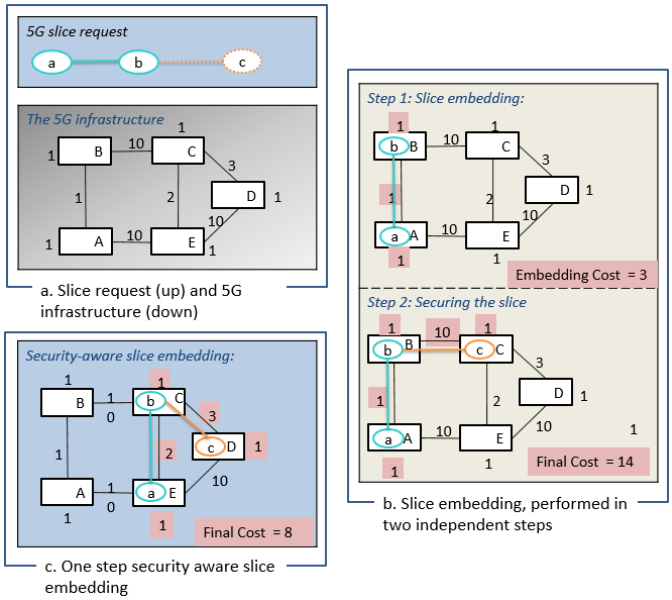


Figure 4: Motivation for designing a security-aware slice request

Figure 4.a gives an example of a slice request composed of two connected VNFs a and b (in blue) and an example of a 5G infrastructure (composed of 5 linked nodes A,B,C,D,E). Assume that each VNF and link requires *one unit* of resources and that each substrate resource has one unit of available resources. The numbers depicted next to each physical resource represent the unit cost (u.c). It is the cost of allocating one unit of physical resource for a virtual component.

Imagine that the traffic outgoing from VNF b should be routed to another VNF c to perform security operations. Consider trying to meet this security need in two ways: either by i) performing two independent steps, i.e. embed the initial slice request and then satisfy the security need, or by ii) considering the security requirements *from the beginning*.

Fig. 4.b illustrates the case where the slice embedding is performed in two independent steps. To embed the initial security request, the less costly physical link $A-B$ is selected for the virtual link $a-b$. The VNFs a and b are mapped to the hosts A and B respectively. The slice embedding cost is then 3 (the sum of mapping $a-b$, a and b). Now, consider handling the security need. To route the traffic outgoing from b, the only available link is $B-C$ and it costs 10, thus VNF c is mapped to C and the virtual link $b-c$ to $B-C$ and this costs 11 units. The total slice embedding cost is 14.

Fig. 4.c illustrates the case where the slice security needs are considered from the beginning. A more optimal embedding solution is found. In fact, assuming that all hosts are free, rather then selecting the physical link $A-B$ to host $a-b$, $C-E$ can be selected. Although more costly, it has the merit of allowing a cheaper allocation of the virtual link $b-c$. The final cost is then 8.

In addition to reducing the embedding cost, considering the security needs from the beginning avoids re-configuring the network. For instance, if the substrate node A, initially hosting a is unsafe, a should be migrated to another physical host. Such reconfigurations can disturb the running service and incur penalty fees.

For these reasons, we add a *preliminary step* before the embedding stage. During this phase, the well known security needs and best practices are modeled and integrated to the initial slice design. The resulting *security-aware slice request* will be fed to the embedding algorithm.

3.3. The security-aware slice design

In this section, we will describe a *basic slice request model* and then detail the different steps leading to a *security-aware slice request model*. But, we will first define the infrastructure model.

3.3.1. The infrastructure model

Without loss of generality, we model the 5G infrastructure by a weighted undirected graph $G_s = (N_s, L_s)$, where N_s is the set of *substrate nodes* n_s and L_s is the set of *substrate links* l_s. The available capacity of node n_s (typically CPU, storage and memory) is denoted by $c(n_s)$ and let $bw(l_s)$ be the available bandwidth on link l_s. Variable φ represents a substrate path (a single or a sequence of substrate links) between two substrate nodes. Variable P_φ is the set of loop-free substrate paths. The available bandwidth $bw(\varphi)$ associated to a substrate path φ can be evaluated as the smallest available bandwidth on the links along the substrate path (see Equation 1).

$$bw(\varphi) = min(\{bw(l_s), \forall l_s \in \varphi\}) \quad (1)$$

3.3.2. Basic slice request model

A slice request r can be represented by a Virtual Network. A virtual network can be modeled by a weighted undirected graph $G_r = (N_r, L_r)$, where N_r is the set of *virtual nodes* representing the VNFs of the slice request and L_r is the set of *virtual links* connecting the VNFs. Each VNF $n_r \in N_r$ requires an amount of resources $c(n_r)$ and each virtual link $l_r \in L_r$ requires an amount of bandwidth $bw(l_r)$.

3.3.3. Securing the slice service request

To secure a service, the universal solution consists in including appropriate security functions (such as firewall, authentication functions, IDS, or DPI) to perform real-time security analysis and operations (detect intrusions, classify the traffic, monitor the flows, guarantee authentication access, etc.) Liu et al.

(2017); Firoozjaei et al. (2017). However, this solution cannot satisfy the security requirements related to resource sharing and placement. For instance, deploying the VNFs in untrusted domains is not permitted for services with high sensitivity. Thus, we conceive a more global security strategy based on two stages: we first include the required virtual network security functions to the service graph and then define constraints related to the resource sharing and location restrictions. Theses stages are detailed below.

Inserting virtual network security functions. To secure the services, additional VNSFs can be included to the service graph. The traffic will traverse these VNSFs where security and monitoring operations will be performed. Note that deciding which VNSFs to insert and where in the slice topology is specific to the service use case and it is generally resolved by security experts after studying the possible threats and vulnerabilities. To each VNSF added to the graph, we associate the amount of required resources and the amount of bandwidth necessary to route the flow to and from the VNSF. Let $G'_r = (N'_r, L'_r)$ denote the graph resulting from this stage. Then, N'_r is an extension of N_r containing the additional VNSF and L'_r contains the virtual links connecting them to the rest of the graph. Using the example of the previous section, $N_r = \{a, b\}$ and $N'_r = \{a, b, c\}$, similarly, $L_r = \{a - b\}$ and $L'_r = \{a - b, b - c\}$.

Defining resource sharing and location constraints. In addition to inserting VNSFs, some security constraints need to be considered during the slice embedding stage, namely: i) those specifying the resource sharing policy, and ii) the constraints related to geographical location restrictions when mapping the slice to the network infrastructure. Motivation and details are given below.

- **The resource sharing constraints** specify whether the same physical host can be shared between different virtual components or no. In other terms, this deals with allowing or forbidding the co-location of VNFs (*resp.* virtual links) in the same physical node (*resp.* physical link). In some scenarios, the co-location of virtual components may be required. For example, to improve the security service performance, a firewall and an IDS can be placed in the same node to collaborate and detect malicious activities within a short time. The co-location may also be required to avoid security breaches (between VNFs of the same service, for example). However, in other scenarios, the co-location of VNFs and links of different services may be forbidden to guarantee isolation between the services, as discussed in Section 3.1.

  To model such constraint, we define for each VNF $n_r \in N'_r$, two sets *Must co-locate* and *Forbid co-location*, denoted $MC(n_r)$ and $FC(n_r)$ respectively. $MC(n_r)$ (*resp.* $FC(n_r)$) contains the list of VNFs that must (*resp.* must not) be allocated in the same host as n_r. Similarly, we specify for each virtual link $l_r \in L'_r$ the sets $MC(l_r)$ (*resp.* $FC(l_r)$) containing the list of virtual links that must (*resp.* must not) share the same *physical path* as l_r. Obviously, a VNF that must be co-located with f must not be in the set $FC(f)$ and vice versa; the same condition is valid for virtual links. Note that the set of VNFs that are neither in $MC(n_r)$ nor in $FC(n_r)$ are simply allowed to be co-located with n_r, the same property is observed for virtual links.

- **The location related constraints** define geographical and domain restrictions when mapping the slice request. In fact, as discussed in Section. 3.1, the 5G infrastructure can be partitioned into *domains of different security levels*. To avoid threats from vulnerable and untrusted hosts or domains, the service network providers can limit the subset of physical resources that may host their service.

  To formulate these constraints, we define for each virtual resource, the set of physical resources allowed to host it, called *May Host*, and denoted MH. Then $MH(n_r) \subset N_s$ is the set of physical nodes authorized to host the VNF n_r, and $MH(l_r) \subset L_s$ is the set of physical links allowed to host the virtual link l_r.

3.3.4. Security-aware service request

In conclusion, a security-aware slice request can be described by a weighted undirected graph $SecG_r = (SecN_r, SecL_r)$ containing the same set of VNFs and links as G'_r and enriched by the following attributes: VNFs security constraints, that is, i) $MC(n_r)$ ii) $FC(n_r)$, iii) $MH(n_r)$, $\forall n_r \in N'_r$; and virtual links security constraints, that is, i) $MC(l_r)$ ii) $FC(l_r)$ iii) $MH(l_r)$. $\forall l_r \in L'_r$. To alleviate the notation, we will reuse the term $G_r = (N_r, L_r)$ to designate the slice request resulting from this step. The notations are summarized in the Appendix.

4. Securing the slice embedding

Once the slice request has been enriched by security constraints, an efficient embedding scheme should be found to allocate and instantiate the resources that will support the service. In this section, we present a security-aware slice embedding solution. To do so, we *first* formulate the problem and then describe the solution.

4.1. System model and problem formulation

Considering the slice request and the 5G infrastructure models introduced in the previous section, the **SA-CNSE** problem consists in finding a mapping $\mathcal{M} : G_r \mapsto G_s$, associating each virtual component of the slice request to a physical host such that the slice resource requirements are satisfied and the different constraints (resource limits, flow conservation and security constraints) are respected.

To formalize the problem, let us introduce two matrixes x and y of binary variables to describe the mapping of VNFs and virtual links, where $x^{n_r}_{n_s} \rightarrow N_s \times N_r$ and $y^{n_r m_r}_{n_s m_s} \rightarrow (L_s)^2 \times (L_r)^2$, defined below:

- VNF mapping:

$$x^{n_r}_{n_s} = \begin{cases} 1, \text{ if VNF } n_r \text{ is mapped to physical node } n_s \\ 0, \text{ else} \end{cases}$$

- Virtual link mapping

$$y_{n_s m_s}^{n_r m_r} = \begin{cases} 1, \text{ if the virtual link } (n_r,m_r) \text{ passes through} \\ \text{the substrate link } (n_s,m_s). \\ 0, \text{ else} \end{cases}$$

The $SA-CNSE$ problem can be formulated as an ILP model as follows:

- **The objective** is to minimize the embedding cost, i.e. the sum of physical resources (per unit cost) spent to satisfy the request:

$$\min \sum_{n_r \in N_r} \sum_{n_s \in N_s} x_{n_s}^{n_r} c(n_r) cost(n_s) + \sum_{(n_r,m_r) \in L_r} \sum_{(n_s,m_s) \in L_s} y_{n_s m_s}^{n_r m_r} bw((n_r,m_r)) cost((n_s,m_s)) \quad (2)$$

where $cost(n_s)$, *resp.* $cost((n_s,m_s))$ are the CPU, *resp. bandwidth* unit cost associated to the substrate node n_s, *resp. substrate link* (n_s,m_s).

- **The basic constraints** are the minimum conditions to obtain a correct mapping solution, regardless of the security needs, namely;

$$\sum_{n_s} x_{n_s}^{n_r} = 1, \forall n_r \in N_r \quad (3)$$

$$\sum_{n_r \in N_r} x_{n_s}^{n_r} c(n_r) \leqslant c(n_s), \forall n_s \in N_s \quad (4)$$

$$\sum_{(n_r,m_r) \in L_r} y_{n_s m_s}^{n_r m_r} bw((n_r,m_r)) \leqslant bw((n_s,m_s)), \quad \forall (n_s,m_s) \in L_s \quad (5)$$

$$\sum_{m_s \in N_s} (y_{n_s m_s}^{n_r m_r} - y_{m_s n_s}^{n_r m_r}) = x_{n_s}^{n_r} - x_{n_s}^{m_r}, \quad \forall (n_r,m_r) \in L_r, n_r < m_r, \forall n_s \in N_s, (n_s,m_s) \in L_s \quad (6)$$

Eq. (3) ensures that each VNF is mapped, and that it is mapped to only one substrate node.

Eq. (4) and Eq. (5) describe constraints on the *node resources*, i.e., CPU, memory and storage, and *bandwidth limitation*, which ensure that the available resources of each physical node and link are not exceeded.

Eq. (6) is the *flow conservation constraint* and ensures that, for each flow steering between two virtual nodes n_r and m_r, for each substrate node $n_s \in N_s$,

  - if n_s **is an intermediate node**, i.e. it does not host any of the flow end nodes n_r and m_r (*formally*, $x_{n_s}^{n_r} = x_{n_s}^{m_r} = 0$), then the total flow incoming to n_s, i.e. $\sum_{m_s \in N_s} (y_{n_s m_s}^{n_r m_r}) \times bw((n_r,m_r))$ is equal to the total flow outgoing from n_s, i.e. $\sum_{m_s \in N_s} (y_{m_s n_s}^{n_r m_r}) \times bw((n_r,m_r))$.
  - If n_s is **the source of the** (n_r,m_r) **flow**, i.e. it hosts n_r (*formally*; $x_{n_s}^{n_r} = 1$), then the total outgoing flow minus the total incoming flow is equal to the required bandwidth, formally equal to $(x_{n_s}^{n_r} - x_{n_s}^{m_r}) \times bw((n_r,m_r)) = bw((n_r,m_r))$.
  - Finally, if n_s **is the sink node**, it hosts m_r (*formally*, $x_{n_s}^{m_r} = 1$), then the total outgoing flow minus the total incoming flow is equal to the negative amount of required bandwidth, formally equal to $(x_{n_s}^{n_r} - x_{n_s}^{m_r}) \times bw((n_r,m_r)) = -bw((n_r,m_r))$.

- **The security related constraints** guarantee that the mapping solution is correct from the security point of view, namely:

$$\sum_{n_s \in N_s \setminus MH(n_r)} x_{n_s}^{n_r} = 0, \forall n_r \in N_r \quad (7)$$

$$\sum_{(n_s,m_s) \in L_s \setminus MH((n_r,m_r))} y_{n_s m_s}^{n_r m_r} = 0, \forall (n_r,m_r) \in L_r \quad (8)$$

$$x_{n_s}^{n_r} = x_{n_s}^{m_r}, \forall m_r \in MC(n_r), n_r \in N_r, n_s \in N_s \quad (9)$$

$$y_{n_s m_s}^{n_r m_r} = y_{n_s m_s}^{o_r p_r}, \quad \forall (o_r,p_r) \in MC((n_r,m_r)), (n_r,m_r) \in L_r, (n_s,m_s) \in L_s \quad (10)$$

$$x_{n_s}^{n_r} \times x_{n_s}^{m_r} = 0, \forall m_r \in FC(n_r), n_r \in N_r, n_s \in N_s \quad (11)$$

$$y_{n_s m_s}^{n_r m_r} \times y_{n_s m_s}^{o_r p_r} = 0, \quad \forall (o_r,p_r) \in FC((n_r,m_r)), (n_r,m_r) \in L_r, (n_s,m_s) \in L_s \quad (12)$$

Eq. (7) models the ***May host*** constraint and guarantees that the untrusted physical nodes i.e. those that do not belong to the MH set ($\{n_s \in N_s \setminus MH(n_r)\}$) cannot host the VNFs of the slice (all the mapping variables $x_{n_s}^{n_r}$ are equal to 0). Eq. (8) models the *May Host* constraint for virtual links in the same way.

Eq. (9) is the ***Must co-locate*** constraint and ensures that the VNFs belonging to the same MC set ($\forall m_r \in MC(n_r)$) have the same mapping result ($x_{n_s}^{n_r} = x_{n_s}^{m_r}$). Eq. (10) models the same constraint for the virtual links.

Similarly, Eq. (11) prohibits the VNFs of the same FC set ($\forall m_r \in FC(n_r)$) from being mapped to the same host ($x_{n_s}^{n_r} \times x_{n_s}^{m_r} = 0, \forall n_s \in N_s$). Eq. (11) defines the same constraints for the virtual links.

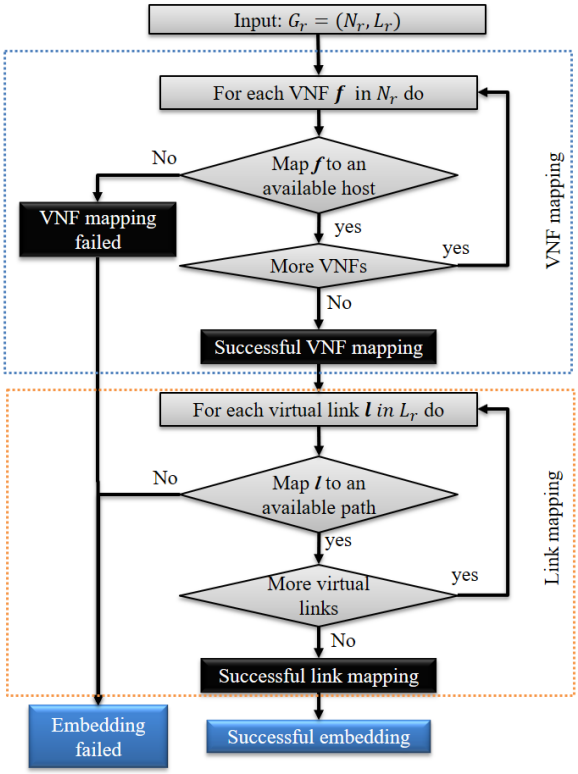


Figure 5: A generic slice embedding algorithm

4.2. Security-aware slice embedding

This section deals with resolving the $SA-CNSE$ problem described above. We demonstrate how non security-aware slice embedding algorithms can be enhanced to handle the security needs while making minimal changes to them. To do so, we will first describe a generic slice embedding algorithm used to benchmark our solution, and then extend it to consider the security constraints.

4.2.1. Generic slice embedding algorithm

As for the bench-marking embedding scheme, we focus on a two-stage slice embedding algorithm (cf. Sect. 2.2.1) depicted in Fig. 5, which is generic and simple enough to describe. The input of the algorithm is a slice request and the output is the mapping result (success or failure). The mapping is performed in two separate steps: a VNF mapping, from N_r to N_s, denoted $\mathcal{M}_N : N_r \mapsto N_s$ and a link mapping from L_r to L_s, denoted $\mathcal{M}_L : L_r \mapsto L_s$. During the VNF mapping, the VNFs are mapped one by one following *a VNF mapping policy* $\mathcal{M}_N$, for example a *greedy policy* Lin et al. (2017), i.e. the VNF with the highest requirements is mapped to the physical node with the most available resources. If one VNF mapping fails, the whole slice mapping fails and the request is rejected. Elsewhere, after mapping all the VNFs, the virtual link mapping is triggered. The virtual links are mapped one by one in a cost effective way, for example using the *shortest path* Gallo & Pallottino (1988) algorithm. The request mapping succeeds only if all the VNFs and links are mapped successfully, the selected resources are then provided to the slice and the 5G infrastructure available resources are updated.

4.2.2. Handling security constraints

In this section, we demonstrate how the security constraints introduced above and defined by the three sets MH, MC and FC can be incorporated into the generic embedding algorithm. To do so, we will focus on the VNF and link mappings separately.

Security-aware VNF mapping. The proposed security-aware VNF mapping algorithm is depicted in the left subfigure of Fig. 6 and described below. Compared to Fig 5, we insert additional steps, highlighted in red, to check the security constraints. In order to map a VNF f, the algorithm first checks the *Must co-locate* constraints. In fact, if at least one VNF in $MC(f)$ is already mapped, f must be mapped to the same host and there is no need to search for alternative candidates. Nonetheless, f can be mapped to the common substrate node, denoted H, only if : i) $c(H) > c(f)$, i.e. H has enough available resources, ii) $H \in MH(f)$, i.e. H is allowed to host f, and iii) $\forall m_r \in FC(f), x_H^{m_r} = 0$, i.e. H does not host any VNF forbidden from co-location with f. In the case where none of the VNFs in $MC(f)$ is already mapped, the algorithm should search available substrate nodes that can host f while satisfying the other security constraints. In more detail, a new limited *research space*, denoted N_s' is defined. N_s' is a subset of N_s excluding the nodes i) that do not belong to $MH(f)$ and ii) those hosting VNFs of the $FC(f)$ set. The VNF mapping policy $\mathcal{M}_N$ of the generic algorithm can be used while changing only the embedding space. Formally, the aim is to find a mapping $\mathcal{M}_N : N_r \mapsto N_s'$.

Security-aware virtual link mapping. The link mapping algorithm is extended in the same way as for VNF mapping (see the right subfigure of Fig. 6). However, a special attention is paid to the fact that a virtual link can be mapped to several substrate links composing the path. Thus, every single substrate link of the whole path should be checked.

Finally, the security-aware slice embedding can be deduced by combining the security-aware VNF and link mappings as in the generic algorithm scheme (cf. Fig. 5).

5. Evaluation and Analysis

The aim of this section is to analyse how satisfying the security constraints will impact the efficiency and behavior of a slice embedding algorithm in terms of acceptance rate, embedding cost, execution time, and substrate resources utilization. Three main issues will be explored:

1) How does security-aware slice embedding algorithms compare to non security-aware algorithms ?

2) Which security constraints (Must co-locate, Forbid co-location, May Host) have the most impact on the efficiency of the embedding algorithm ?

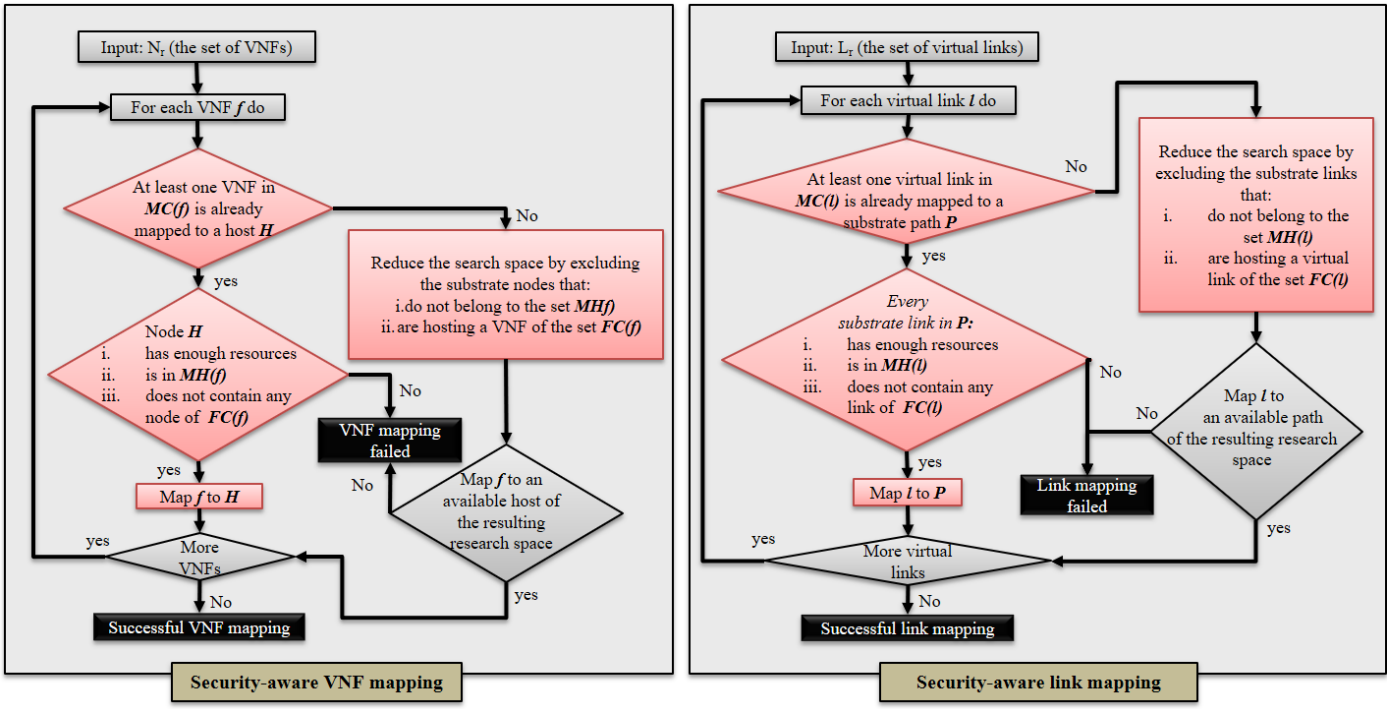


Figure 6: security-aware slice embedding

3) For two-stage embedding algorithms, does satisfying security constraints differ in terms of efficiency between VNF embedding and virtual link embedding?

In order to answer these questions, extensive experiments were conducted in a simulation environment.

5.1. Evaluation environment

We start by describing, in detail, the evaluation environment before presenting and discussing the experimental results. This includes i) the simulation setting, ii) the evaluation metrics, iii) the compared algorithms, and iv) the security constraints generation. Note that the key notations used in this section are summarized in Appendix A.

5.1.1. *Simulation settings*

To carry out the experiments, we have implemented a C++ simulator [1] describing the slice embedding environment. We used the GT-ITM tool Zegura et al. (1996) to generate the network topologies for the 5G infrastructure and service requests following the simulation settings mostly used in the literature Chowdhury et al. (2012); Fischer et al. (2013); Zhang et al. (2018). In more detail, we consider a substrate network composed of 75 substrate nodes where each pair of substrate nodes is randomly connected with probability 0.5. The CPU and bandwidth resources of the substrate nodes and links are real numbers uniformly distributed between 20 and 70.

We also consider 50 service requests. For each request, the number of VNFs in each request is randomly distributed between 2 and 10. Each pair of VNFs is randomly connected with probability 0.5. The CPU requirements of the VNFs are real numbers uniformly distributed between 0 and 20, and the bandwidth requirements of the virtual links are uniformly distributed between 0 and 50.

5.1.2. *Evaluation metrics*

We use state of the art most common performance metrics Cao et al. (2019b); Laghrissi & Taleb (2019), to compare and measure the performance of the embedding algorithms, namely:

- $\mathbb{A}$ is the rate of the service graph requests that have been successfully embedded.
- $\mathbb{RC}$ measures the average revenue to cost ratio for accepted requests. $\mathbb{RC}$ is calculated as follows:

$$\mathbb{RC} = \frac{\sum_{G_r \in \mathcal{AR}} \mathbb{R}(G_r)}{\sum_{G_r \in \mathcal{AR}} \mathbb{C}(G_r)} \tag{13}$$

where $\mathcal{AR}$ is the set of accepted requests, $\mathbb{R}(G_r)$ and $\mathbb{C}(G_r)$ are the request revenue and cost classically defined in the literature Chowdhury et al. (2012).

$$\mathbb{R}(G_r) = \left(\sum_{n_r \in N_r} c(n_r)\right) + \left(\sum_{l_r \in L_r} bw(n_r)\right) \tag{14}$$

The revenue represents the sum of the resources (computing and bandwidth) demanded by the request (see Eq. (14)). The cost is the sum of physical resources (per unit cost) spent to satisfy the request, formally defined in Eq. (2).

[1] The simulator will be publicly released later

- $\mathbb{T}$, the execution time, is the average time needed to process a service graph request regardless of the result (accepted or rejected).

- $\mathbb{ns}$ and $\mathbb{ls}$ are the average substrate nodes and links stress respectively. In order to quantify the resource usage of the substrate network, we use the notion of *stress* Chowdhury et al. (2012). The stress of a substrate node n_s, denoted $stress(n_s)$, is defined as the total amount of resources allocated to all the VNFs hosted on n_s, formally;

$$stress(n_s) = \sum_{n_r \in \mathcal{R}_r} x_{n_s}^{n_r} \times c(n_r) \quad (15)$$

$\mathbb{ns}$ is the *average* node stress and is calculated by averaging the stress of all the substrate nodes. $\mathbb{ls}$ is calculated in the same manner for substrate links.

5.1.3. *Algorithm comparison*

We use the generic embedding scheme, described in Sect. 4.2.1 to devise two embedding algorithms, *BestEmbed* and *WorstEmbed*. Both of them perform VNF and link mappings separately but use different VNF and link mapping methods, as described in Table 1.

Table 1: The compared embedding algorithms description

Algorithm	VNF mapping method	Link mapping method
BestEmbed	Starts by embedding the VNF with the **highest** resource need and maps it to the physical node with the **most** available resources.	Starts by embedding the virtual link with the **highest** bandwidth need and maps it to the path with the **most** available bandwidth.
WorstEmbed	Starts by embedding the VNF with the **lowest** resource need and maps it to the physical node with the **least** available resources.	Starts by embedding the virtual link with **highest** bandwidth need and maps it to the path with the **least** available bandwidth.
SecBestEmbed	The security-aware version of **BestEmbed**	
SecWorstEmbed	The security-aware version of **WorstEmbed**	

Note that *WorstEmbed* leads to a rapid resource saturation and is therefore less efficient than *BestEmbed*. This is confirmed by experimental results (cf. Table 2) showing the superiority of the *BestEmbed* algorithm for all evaluation metrics (more acceptance rate and revenue to cost ratio, lower execution time and substrate network utilization).

Table 2: Performance of non security-aware algorithms

Algorithm	$\mathbb{A}$ (%)	$\mathbb{RC}$	$\mathbb{T}(s)$	$\mathbb{ns}$	$\mathbb{ls}$
BestEmbed	76	0.37	1.32	0.1306	0.0861
WorstEmbed	72	0.25	3.06	0.1385	0.1122

These two algorithms are then extended (following the process described in Sect. 4.2.2) to handle the security constraints. The obtained algorithms are denoted *SecBestEmbed* and *SecWorstEmbed*. The embedding algorithms that will be compared are then 1. *BestEmbed*, 2. *WorstEmbed*, 3. *SecBestEmbed*, and 4. *SecWorstEmbed*.

5.1.4. *The security constraints generation*

To evaluate the security-aware embedding algorithms, security-aware slice requests should be generated. We remind the reader that this is achieved in two steps: first by inserting VNSFs in the request topology, and second by including the resource sharing and location constraints. The first step simply extends the request topology and the output is a larger slice request. Since we simulate a variety of slice requests typologies, of different sizes, they can already represent such output. Thus, we ignore simulating this step and focus on the second stage, i.e. generating the security constraint sets *MC*, *FC* and *MH* described in Sect. 3.

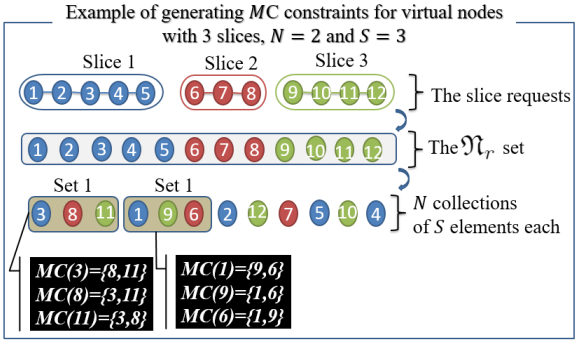


Figure 7: *MC* and *FC* security constraints generation.

Let $\mathfrak{N}_r$ *(resp.* $\mathfrak{L}_r$*)* be the set of VNFs (*resp. links*) existing in all the slice requests. The aim is to generate, for each VNF $n_r \in \mathfrak{N}_r$ (*resp. link* $l_r \in \mathfrak{L}_r$), three sets: i) the VNFs (*resp. virtual links*) that must be co-located with it, i.e *MC*; ii) the VNFs (*resp. virtual links*) forbidden from co-location, i.e. *FC*; and iii) the trusted physical hosts, i.e. *MH*.

To do so, we first initialize all the security constraint sets to `Empty`. Second, for each constraint type, we select $\mathcal{N}$ VNFs (*resp. virtual links*) and fill in their constraint set with $\mathcal{S}$ VNFs (*resp. virtual links*) ($\mathcal{S}$ for set size).

In more detail, to generate the *MH* constraints for VNFs (*resp. virtual links*), we randomly select $\mathcal{N}$ VNFs (*resp. virtual links*) from $\mathfrak{N}_r$ (*resp.* $\mathfrak{L}_r$), and for each selected element, we randomly pick $\mathcal{S}$ substrate nodes (*resp. links*) from N_s (*resp.* L_s) to fill the *MH* set. Note that the values of $\mathcal{N}$ can theoretically range from 1 to $|\mathfrak{N}_r|$ for VNFs and from 1 to $|\mathfrak{L}_r|$ for virtual links, and those of $\mathcal{S}$ from 1 to $|N_s|$ for VNFs and from 1 to $|L_s|$ for virtual links where $|.|$ represents the dimension of the set.

As for *MC* and *FC* sets, they are generated by randomly forming $\mathcal{N}$ disjoint collections of $\mathcal{S}$ VNFs (resp. virtual links) of the $\mathfrak{N}_r$ (resp. $\mathfrak{L}_r$) set as illustrated in the example of Fig. 7. Note that the values of $\mathcal{N}$ and $\mathcal{S}$ are dependent, in particular $\mathcal{N} \times \mathcal{S}$ must not exceed $|\mathfrak{N}_r|$ for VNFs, and $|\mathfrak{L}_r|$ for virtual links.

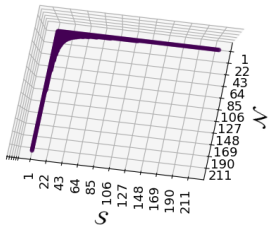
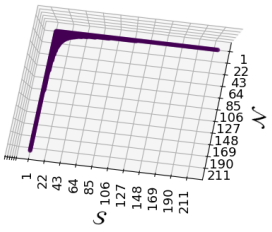
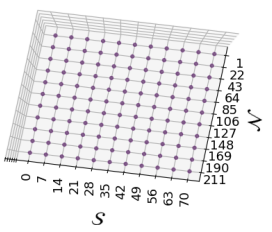
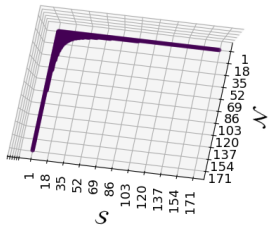
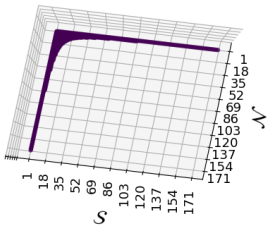
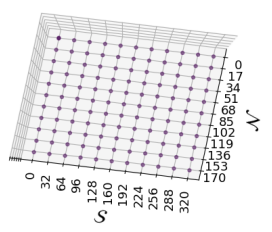
Constraint: Defined for	Must Co-locate (*MC*)	Forbid Co-location (*FC*)	May host (*MH*)
VNFs	$\mathcal{N} \times \mathcal{S} \leqslant \lvert\mathfrak{N}_r\rvert$ 	$\mathcal{N} \times \mathcal{S} \leqslant \lvert\mathfrak{N}_r\rvert$ 	$\mathcal{N} \leqslant \lvert\mathfrak{N}_r\rvert, \mathcal{S} \leqslant \lvert N_s\rvert$ 
Virtual links	$\mathcal{N} \times \mathcal{S} \leqslant \lvert\mathfrak{N}_r\rvert$ 	$\mathcal{N} \times \mathcal{S} \leqslant \lvert\mathfrak{N}_r\rvert$ 	$\mathcal{N} \leqslant \lvert\mathfrak{L}_r\rvert, \mathcal{S} \leqslant \lvert L_s\rvert$ 

Table 3: Evaluation scenarios

5.2. Evaluation results

The aim of the evaluation is to analyse the effects of *taking into account the security constraints* during the slice embedding. To do so, we measure the performance of the embedding algorithms, defined in Section 5.1.3, when managing different security constraints in different scenarios.

In more detail, we define 6 evaluation scenarios, depicted in Table 3, each focusing on one security constraint defined either for VNFs or virtual links. Moreover, for each scenario, we explore the algorithms behavior for a large range of *theoretically* possible values of $\mathcal{N}$ and $\mathcal{S}$. Namely, for *MC* and *FC* scenarios, all the possible values of $\mathcal{N}$ and $\mathcal{S}$, defined by $\mathcal{N} \times \mathcal{S} \leqslant |\mathfrak{N}_r|$ are tested. The area delimited by a rectangular hyperbola curve represent these values. As for the *May host* constraint, a grid of 10×10 uniformly distributed samples of $\mathcal{N}$ and $\mathcal{S}$ values are explored. Using the evaluation settings described in Sect. 5.1.1, we obtained $|\mathfrak{N}_r| = 217$ and $|\mathfrak{L}_r| = 171$. Note that some values of $\mathcal{N}$ and $\mathcal{S}$ are unrealistic but will be explored to measure the theoretical scalability limits of the proposed solution. For example, in the evaluation scenario where *MC* constraints are defined for the VNFs (illustrated by the cell at the first row and first column of Table 3): $\mathfrak{N}_r = 211$ means that *each VNF must be co-located with $\mathcal{S}$ other VNFs*. This does not correspond to a probable scenario, but we will investigate it to study the behaviour of the algorithm at the border of realistic scenarios. In general, realistic scenarios correspond to low values of $\mathcal{N}$ and $\mathcal{S}$.

To evaluate *SecBestEmbed* (resp. *SecWorstEmbed*), we depict, for each evaluation metric, the difference between the value obtained by *SecBestEmbed* and *BestEmbed* (resp. *SecWorstEmbed* and *WorstEmbed*). When the difference is positive, values are drawn in graduations of red, elsewhere, graduations of blue are used.

The results are reported in Tables 4,5 and 6 for *Must Co-locate*, *Forbid co-location* and *May Host* constraints respectively. All figures can be viewed with higher resolution, and from multiple angles using the link below[2]. All the reported results are obtained by averaging the collected performance from 10 independent runs for each simulation point. For each run, new security constraint sets are randomly generated. All tests were conducted under the same conditions on the same machine, running an Intel(R) Core(TM)-i7-7920HQ 3.10 GHz processor with 64 GB RAM.

5.3. Analysis and discussion

In this section, we analyse the experimental results and answer the questions raised in Sect. 5.

5.3.1. *How does security-aware slice embedding algorithms compare to non security-aware algorithms ?*

Acceptance rate & substrate network utilization. The subfigures on the first line of each table depict the acceptance rates obtained for different $\mathcal{N}$ and $\mathcal{S}$ values. Similarly, the subfigures of the second *(resp. third)* rows of each table show the substrate node *(resp. link)* average stress. Comparing the above rows, we notice that the three metrics ($\mathbb{A}$, $\mathbb{ns}$ and $\mathbb{ls}$) have similar behaviour. This is expected as the substrate network usage is heavily correlated to the acceptance rate. In fact, the more requests are accepted, the more substrate resources are used to embed them and thus the more $\mathbb{ns}$ and $\mathbb{ls}$ are high, and vice versa. For this reason, these metrics will be interpreted together. Looking at all the subfigures, we notice that $\mathbb{A}$, $\mathbb{ns}$ and $\mathbb{ls}$ decrease more or less rapidly when $\mathcal{N}$ and $\mathcal{S}$ increase, especially for *Must co-locate* and *Forbid co-location* constraints. In

[2] https://gitlab.com/security_aware_slice_embedding/security_aware_slice_embedding.git

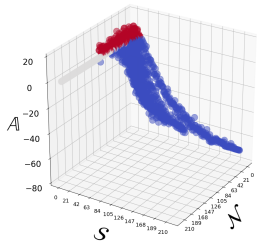
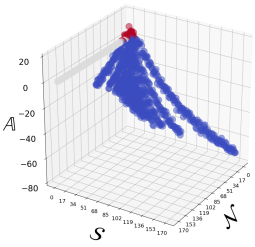
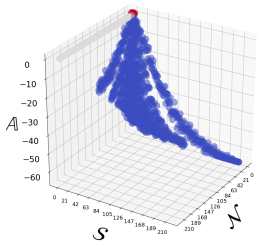
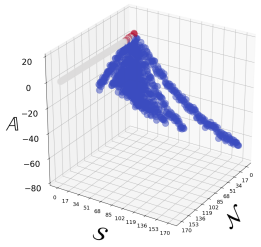
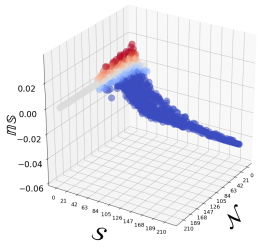
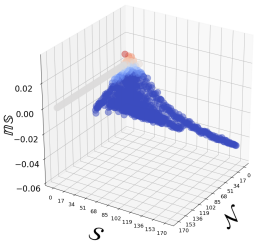
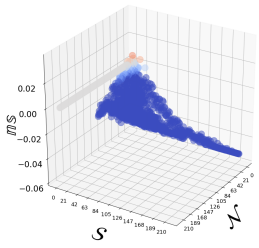
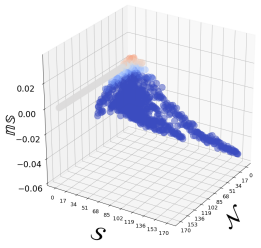
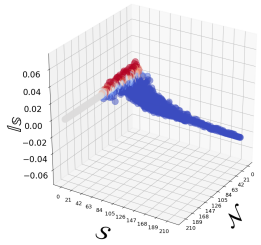
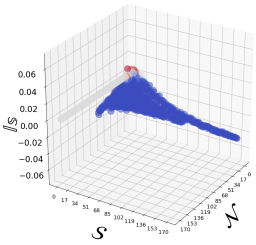
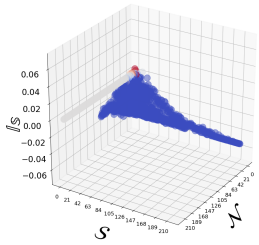
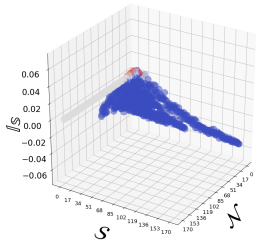
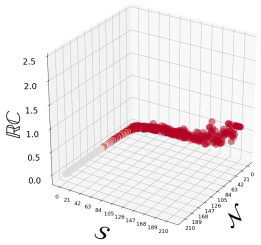
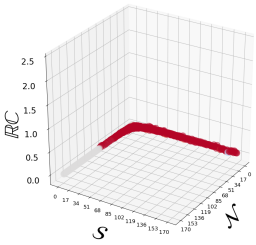
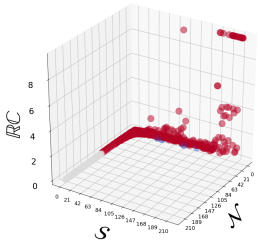
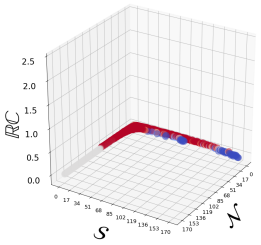
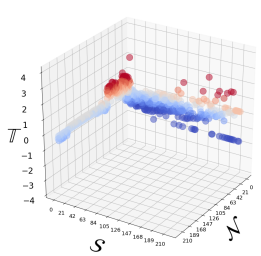
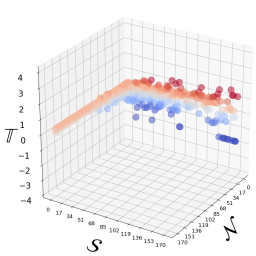
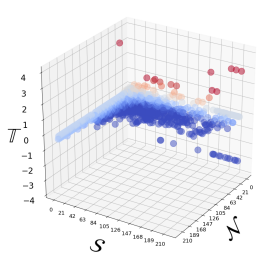
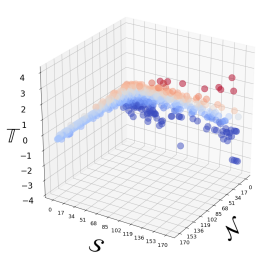
	SecBesttEmbed		***SecWorstEmbed***	
	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links
$\mathbb{A}$	subFig 1.A	subFig 1.B	subFig1.C	subFig1.D
				
Notation : The red color means that **more slice requests are accepted**, compared to non security-aware algorithms				
$\mathbb{ns}$	subFig2.A	subFig2.B	subFig2.C	subFig2.D
				
Notation : The red color means that more the substrate nodes are **more stressed**, compared to non security-aware algorithms				
$\mathbb{ls}$	subFig3.A	subFig3.B	subFig3.C	subFig3.D
				
Notation : The red color means that more the substrate links are **more stressed**, compared to non security-aware algorithms				
$\mathbb{RC}$	subFig4.A	subFig4.B	subFig4.C	subFig5.D
				
Notation : The red color means that the embedding is **more costly**, compared to non security-aware algorithms				
$\mathbb{T}$	subFig5.A	subFig5.B	subFig5.C	subFig5.D
				
Notation : The red color means that the embedding algorithm is **more time consuming**, compared to non security-aware algorithms				

Table 4: Evaluation results: impact of the *MC* constraints over the slice embedding process

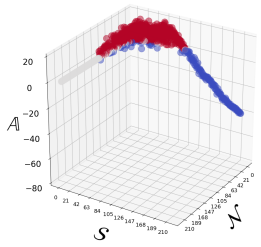
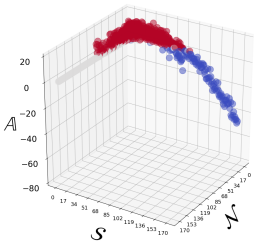
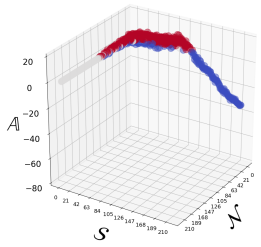
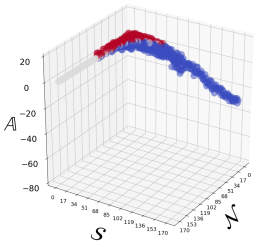
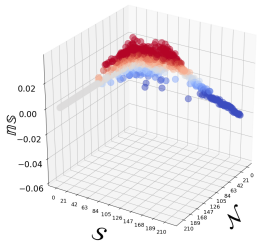
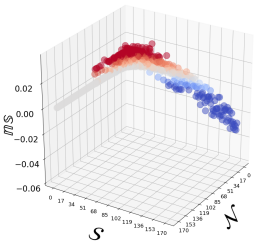
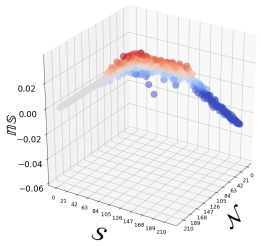
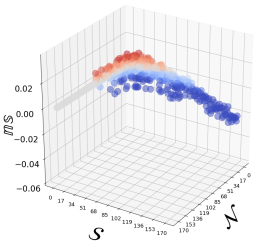
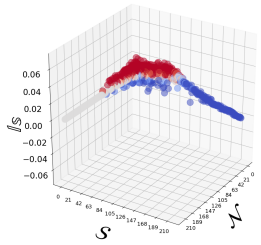
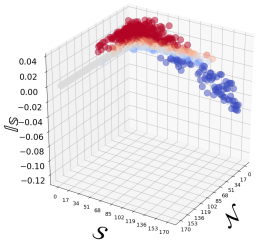
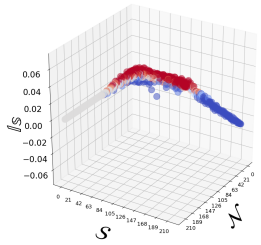
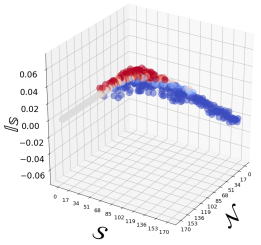
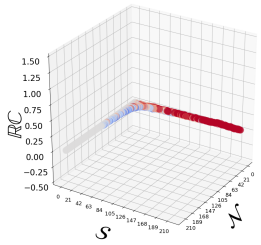
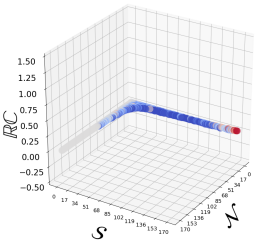
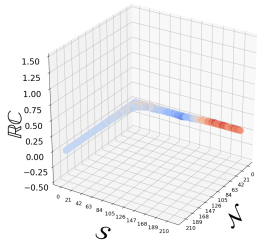
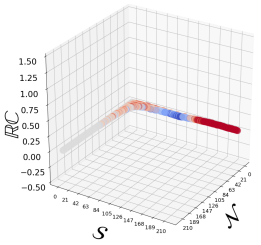
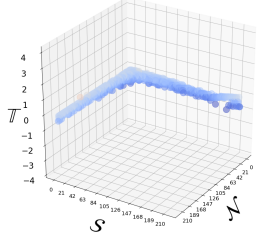
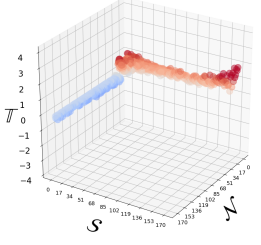
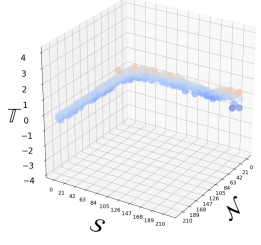
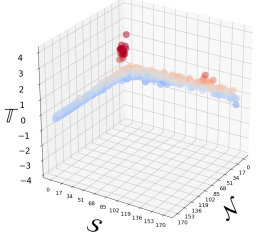
	SecBesttEmbed		***SecWorstEmbed***	
	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links
$\mathbb{A}$	subFig 1.A	subFig 1.B	subFig1.C	subFig1.D
				
Notation : The red color means that **more slice requests are accepted**, compared to non security-aware algorithms				
$\mathbb{ns}$	subFig2.A	subFig2.B	subFig2.C	subFig2.D
				
Notation : The red color means that more the substrate nodes are **more stressed**, compared to non security-aware algorithms				
$\mathbb{ls}$	subFig3.A	subFig3.B	subFig3.C	subFig3.D
				
Notation : The red color means that more the substrate links are **more stressed**, compared to non security-aware algorithms				
$\mathbb{RC}$	subFig4.A	subFig4.B	subFig4.C	subFig5.D
				
Notation : The red color means that the embedding is **more costly**, compared to non security-aware algorithms				
$\mathbb{T}$	subFig5.A	subFig5.B	subFig5.C	subFig5.D
				
Notation : The red color means that the embedding algorithm is **more time consuming**, compared to non security-aware algorithms				

Table 5: Evaluation results: impact of the *FC* constraints over the slice embedding process

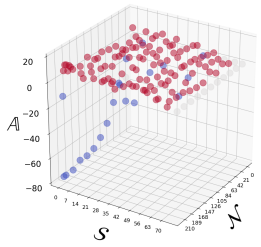
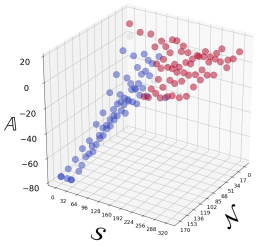
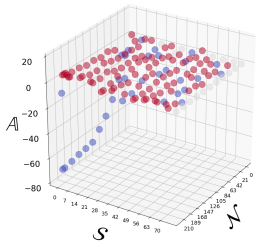
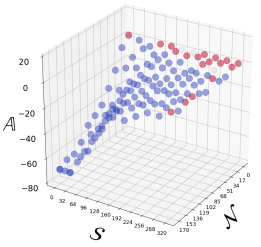
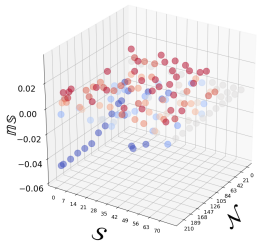
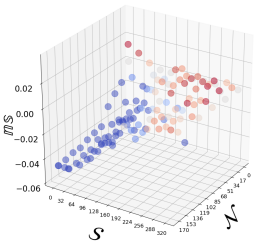
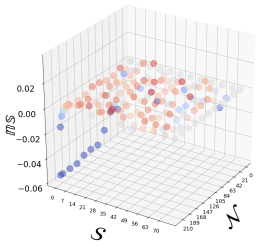
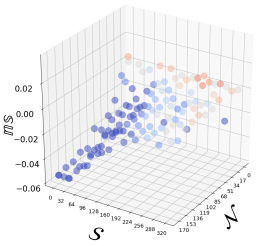
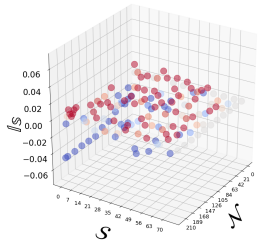
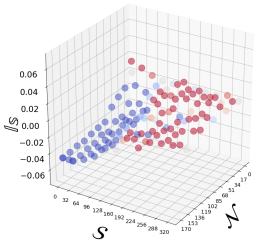
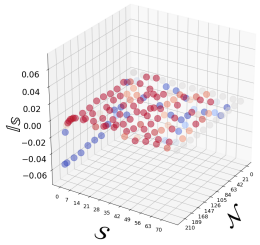
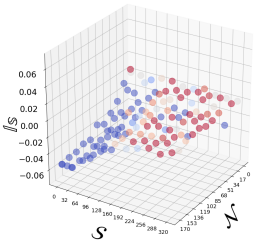
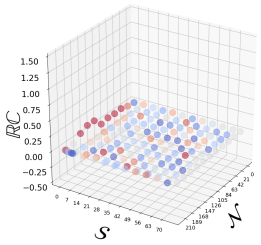
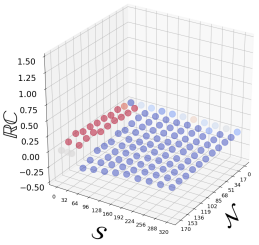
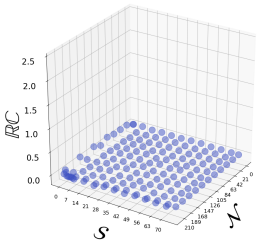
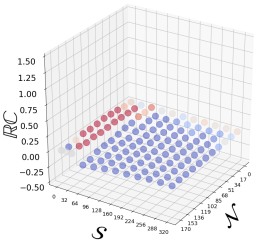
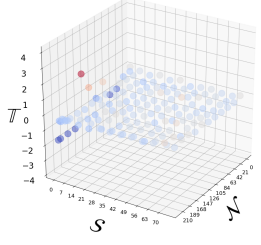
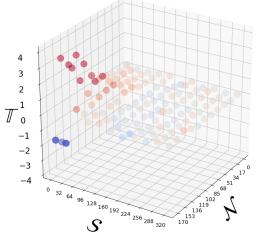
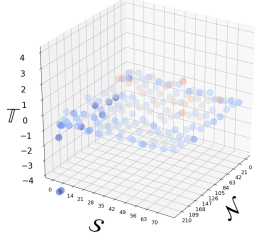
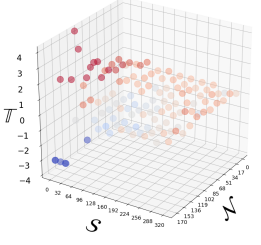
	SecBesttEmbed		***SecWorstEmbed***	
	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links	Must Co-locate (*MC*) for VNFs	Must Co-locate (*MC*) for links
$\mathbb{A}$	subFig 1.A	subFig 1.B	subFig1.C	subFig1.D
				
Notation : The red color means that **more slice requests are accepted**, compared to non security-aware algorithms				
$\mathbb{ns}$	subFig2.A	subFig2.B	subFig2.C	subFig2.D
				
Notation : The red color means that more the substrate nodes are **more stressed**, compared to non security-aware algorithms				
$\mathbb{ls}$	subFig3.A	subFig3.B	subFig3.C	subFig3.D
				
Notation : The red color means that more the substrate links are **more stressed**, compared to non security-aware algorithms				
$\mathbb{RC}$	subFig4.A	subFig4.B	subFig4.C	subFig5.D
				
Notation : The red color means that the embedding is **more costly**, compared to non security-aware algorithms				
$\mathbb{T}$	subFig5.A	subFig5.B	subFig5.C	subFig5.D
				
Notation : The red color means that the embedding algorithm is **more time consuming**, compared to non security-aware algorithms				

Table 6: Evaluation results: impact of the *MH* constraints over the slice embedding process

addition, for low values of $\mathcal{N}$ and $\mathcal{S}$, *i.e for a small numbers of constraints*, $\mathbb{A}$,$\mathbb{ns}$ and $\mathbb{ls}$ remain high, *i.e. the embedding algorithms manage to satisfy the requests but lead to more substrate network utilization. On the other case, the more constraints there are, the more difficult it is to manage them, and the less stressed is the substrate network.* Moreover, by comparing the subfigures representing $\mathbb{A}$ for the *BestEmbed* algorithm (subFig1.A and subFig1.B, in all the tables) to those of *WorstEmbed* (subFig1.C and subFig1.D) , we notice that $\mathbb{A}$ decreases more rapidly for the *WorstEmbed* algorithm (there are more dots in red in the subFig1.A and subFig1.B). *Thus, a security-aware algorithm derived from an efficient non security-aware algorithm, manages to accept more security-aware slice requests than one derived from a less efficient algorithm, but naturally lead to a more stressed substrate network (looking at the $\mathbb{ns}$ and $\mathbb{ls}$ Figures).*

Revenue Cost. The subFigures showing $\mathbb{RC}$ for the different scenarios are depicted in the fourth row of each table. We note that $\mathbb{RC}$ varies slightly depending on the values of $\mathcal{N}$ and $\mathcal{S}$ and that the variation (positive or negative, i.e. red or blue) depends on the security type. Thus, *handling the security constraints does not necessarily introduce an over cost compared to non security-aware algorithms.*

Execution Time. The subfigures showing the execution time for the different scenarios are in the last row of each table. Note that contrary to the previous metrics, that is, $\mathbb{A}$ and $\mathbb{RC}$, the blue color reflects an improvement in the time performance of the algorithm, that is, a decrease in the execution time. We note that $\mathbb{T}$ varies slightly depending on the values of $\mathcal{N}$ and $\mathcal{S}$. Moreover, the variation is not regular. *We conclude that handling the security constraints does not necessarily increase the request processing time.*

5.3.2. *Which security constraints have the more impact on the efficiency of the embedding algorithm ?*

Acceptance rate & substrate network utilization. Looking at the subfigures representing $\mathbb{A}$, $\mathbb{ns}$ and $\mathbb{ls}$, we remark that the number of constraints $\mathcal{N}$ has more influence than their size $\mathcal{S}$ for *MC* and *FC* constraints, unlike the *MH* constraint where it is the size of the constraints that matters. Moreover, by comparing $\mathbb{A}$, $\mathbb{ns}$ and $\mathbb{ls}$ for the *FC* constraint with those of the *MC* constraint, we can see that there are more positive values, denoted by red dots, and fewer negative values, denoted by blue dots, *so the Forbid co-location constraint is easier to accommodate than the Must co-locate constraint but leads to higher node and link stress rates*. Indeed, the *MC* constraint is more restrictive since it requires the co-location of several virtual elements in the same host, which is quickly limited by the size of the host. Naturally, handling *FC*, leads to higher acceptance rate and thus a more stressed substrate network, compared to managing the *MC* constraints. Compared to *MC* and *FC*, the acceptance rate when handling the *MH* constraints is higher and the substrate network is more stressed. *May host is the easiest constraint to manage, but leads to a more saturated substrate network*. In fact, for the *MC* constraint, apart from the common host (described in Sect. 4.2.2), all the other hosts are untrusted, which is very constraining. Similarly, for the *FC* constraint, all the substrate nodes hosting VNFs (or links) in the *FC* set are ineligible.

Revenue Cost. Comparing the Revenue Cost for different constraint types, we notice that managing the *Must co-locate* constraint is the most costly embedding while handling the *May co-locate* is the less costly. This may be explained by the same arguments given above, in fact, the *MC* constraint is the most restrictive one while the *MH* constraints are the easiest to handle.

Execution Time. For *MC* constraints, $\mathbb{T}$ slightly decreases as the number of constraints $\mathcal{N}$ increases. *Handling more Must co-locate constraints improves the execution time.* This can be explained by the fact that the algorithm will frequently ($\mathcal{N}$ times) process less steps, since it will only have to check if the common host has enough resources instead of identifying all potential hosts in the substrate network (compare Fig. 6 and Fig. 5), which can be time consuming. For the *May host* constraints, there is a slight decrease of $\mathbb{T}$ for most of the $\mathcal{N}$ and $\mathcal{S}$ values. *Handling May host constraints improves the execution time*. This is expected since this constraint reduces the research space size. In fact, for each virtual component (VNFs and links) in the slice request, the embedding algorithm searches for potential hosts (with enough available resources) only in the subsets of legitimate hosts defined by the *May host* constraint, associated to each virtual component, instead of going through all the substrate nodes and links, which is time consuming. For the *FC* constraint, $\mathbb{T}$ fluctuates slightly with $\mathcal{N}$ and $\mathcal{S}$, but remains comparable to the values obtained by the non security-aware algorithm.

5.3.3. *Does satisfying security constraints differ in terms of efficiency between VNF embedding and virtual link embedding?*

Acceptance rate & substrate network utilization. By comparing $\mathbb{A}$,$\mathbb{ns}$ and $\mathbb{ls}$ for *MC* and *FC* constraints, we can notice that there are more red dots when the constraint is defined for VNFs. *So it is easier to satisfy security constraints when they are defined for VNFs than for links for the MC and FC constraints.* The same behavior can be observed for the *MH* constraint, where $\mathbb{A}$ drops for the case where $\mathcal{N}$ increases while $\mathcal{S}$ decreases. This represents the case where the subset of legitimate hosts (belonging to the *MH* sets) decreases for an increasing number of VNFs. However, when the constraint is defined for links, the decrease is more noticeable. *Therefore, for all the constraint types, it is easier to handle the constraints when they are defined for VNFs than for links.* This is due to the fact that a virtual link l_r can be hosted by a path composed of several physical links (see Sect. 4.2.2). Therefore, excluding a physical link from the set of eligible hosts associated to l_r may result in the exclusion of several possible paths, which will in turn decrease the chance of finding an available path to host l_r and lead to request rejection.

Revenue Cost. Looking at the subFigures depicting the cost for the *MC* constraint, we notice a slight improvement in $\mathbb{RC}$ when $\mathcal{S}$ increases, that is, the case where large collections of VNFs are co-located. The drop in $\mathbb{RC}$ can be explained by the fact that if several VNFs are co-located, the cost of allocating links to connect them is null since they are already in the same host, which decreases the slice embedding cost and thus increases $\mathbb{RC}$. *But, in general, satisfying constraints when they are defined for VNFs or for links has approximately the same cost.*

Execution Time. Looking at the different subfigures plotting $\mathbb{T}$, we note that the variation of $\mathbb{T}$ is comparable when the constraints are defined for the VNFs or links.

5.4. Discussion

In this section, we discuss the strengths and weaknesses of our solution. The major advantage of $SA-CNSE$ is its ability to model and satisfy different security constraints to prevent the threats described in Section 3.1. This is essential to fully enjoy the benefits of 5G slicing without compromising the security of the running services. Moreover, the simulation results showed that, the security-aware slice embedding algorithm performance vary depending on the number ($\mathcal{N}$) and size ($\mathcal{S}$) of the handled constraint as well as its type (*MC*, *FC* or *MH*). In addition, a security aware embedding algorithm derived from an efficient non security-aware embedding algorithm achieves better results, then a one derived from a less efficient embedding algorithm. Compared to non security-aware solutions, our model manages to handle a reasonable number and size of constraints (low values of $\mathcal{N}$ and $\mathcal{S}$), with high acceptance rate and without automatically introducing an over cost or extra processing time. However, meeting some restrictive security constraints (namely the *Must co-locate* constraints) may be costly, and can saturate the network. Thus a trade-off between these conflicting objectives, that is i) secure the slice on one hand, and ii) improve the embedding performance metrics on the other hand, should be considered. Finally, we showed how *two-stage* embedding algorithms can be improved to take into account security needs, but adapting the solution to *one-stage* embedding algorithms may require more investigation.

6. Conclusion

While the 5G slice resource allocation problem was extensively investigated in the literature, little attention was paid to the security aspects. To fill this gap, in this paper, we proposed a *security-aware 5G core network slice embedding*, solution. The aim is to: i) model and ii) meet the slice security needs to avoid cyber attacks. To achieve this, we followed a security-by-design approach in two steps. First, we proposed a security-aware slice request to describe the security needs. Second, we solved the slice embedding problem while respecting the security constraints. Through extensive experiments, we evaluated the scalability of handling the security constraints in terms of acceptance rate, substrate network usage, cost to revenue ratio and execution time. The results showed that, compared to non security-aware algorithms, our solution manages to handle reasonable number of constraints without automatically introducing an over cost or extra processing time. In the future work, we plan to investigate the security-aware slice embedding for one stage, distributed and multi-domain algorithms. Moreover, an exact solution for the security-aware embedding problem based on the ILP formalization, will be devised. Finally, we aim to extend the security-aware slice design to model other security constraints.

References

Arfaoui, G., Bisson, P., Blom, R., Borgaonkar, R., Englund, H., Félix, E., Klaedtke, F., Nakarmi, P. K., Näslund, M., O'Hanlon, P., Papay, J., Suomalainen, J., Surridge, M., Wary, J., & Zahariev, A. (2018). A security architecture for 5g networks. *IEEE Access*, *6*, 22466–22479.

Bays, L. R., Oliveira, R. R., Buriol, L. S., Barcellos, M. P., & Gaspary, L. P. (2012). Security-aware optimal resource allocation for virtual network embedding. In *Proceedings of the 8th International Conference on Network and Service Management* (pp. 378–384). International Federation for Information Processing. doi:10.13140/2.1.3170.6569.

Bays, L. R., Oliveira, R. R., Buriol, L. S., Barcellos, M. P., & Gaspary, L. P. (2014). A heuristic-based algorithm for privacy-oriented virtual network embedding. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (pp. 1–8). doi:10.1109/NOMS.2014.6838360.

Bhamare, D., Jain, R., Samaka, M., & Erbad, A. (2016). A survey on service function chaining. (pp. 138 – 155). volume 75. doi:10.1016/j.jnca.2016.09.001.

Boutigny, F., Betgé-Brezetz, S., Blanc, G., Lavignotte, A., Debar, H., & Jmila, H. (2020). Solving security constraints for 5G slice embedding: A proof-of-concept. *Computers & Security*, *89*, 101662. doi:10.1016/j.cose.2019.101662.

Cao, H., Wu, S., Guo, Y., Zhu, H., & Yang, L. (2019a). Mapping strategy for virtual networks in one stage. *IET Communications*, *13*, 2207–2215. doi:10.1049/iet-com.2018.6175.

Cao, H., Wu, S., Hu, Y., Liu, Y., & Yang, L. (2019b). A survey of embedding algorithm for virtual network embedding. *China Communications*, *16*, 1–33.

Cao, H., Zhu, H., & Yang, L. (2020). Collaborative attributes and resources for single-stage virtual network mapping in network virtualization. *Journal of Communications and Networks*, *22*, 61–71. doi:10.1109/JCN.2019.000045.

Cao, H., Zhu, Y., Yang, L., & Zheng, G. (2017). A efficient mapping algorithm with novel node-ranking approach for embedding virtual networks. *IEEE Access*, *5*. doi:10.1109/ACCESS.2017.2761840.

Chowdhury, M., Rahman, M. R., & Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, *20*, 206 – 219. doi:10.1109/TNET.2011.2159308.

Chowdhury, N. M. K., & Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, *54*, 862 – 876. doi:10.1016/j.comnet.2009.10.017.

Dolati, M., Hassanpour, S. B., Ghaderi, M., & Khonsari, A. (2019). Deepvine: Virtual network embedding with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 879–885). IEEE. doi:10.1109/INFCOMW.2019.8845171.

Doriguzzi-Corin, R., Scott-Hayward, S., Siracusa, D., & Salvadori, E. (2017). Application-centric provisioning of virtual security network functions. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (pp. 276–279). doi:10.1109/NFV-SDN.2017.8169861.

Doriguzzi-Corin, R., Scott-Hayward, S., Siracusa, D., Savi, M., & Salvadori, E. (2019). Dynamic and application-aware provisioning of chained virtual security network functions. *CoRR*, *abs/1901.01704*. doi:10.1109/TNSM.2019.2941128. arXiv:1901.01704.

Esposito, F., Di Paola, D., & Matta, I. (2013). A general distributed approach to slice embedding with guarantees. In *2013 IFIP Networking Conference* (pp. 1–9).

Esposito, F., Di Paola, D., & Matta, I. (2016). On distributed virtual network embedding with guarantees. *IEEE/ACM Transactions on Networking (TON)*, *24*, 569–582. doi:10.1109/TNET.2014.2375826.

Firoozjaei, M. D., Jeong, J. P., Ko, H., & Kim, H. (2017). Security challenges with network functions virtualization. *Future Generation Computer Systems*, *67*, 315–324. doi:10.1016/j.future.2016.07.002.

Fischer, A., Botero, J. F., Beck, M. T., de Meer, H., & Hesselbach, X. (2013). Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, *15*, 1888–1906. doi:10.1109/SURV.2013.013013.00155.

Gallo, G., & Pallottino, S. (1988). Shortest path algorithms. *Annals of operations research*, *13*, 1–79. doi:10.1007/BF02288320.

Guan, J., Wei, Z., & You, I. (2018). GRBC-based network security functions placement scheme in SDS for 5G security. *Journal of Network and Computer Applications*, . doi:10.1016/j.jnca.2018.03.013.

He, M., Zhuang, L., Tian, S., Wang, G., & Zhang, K. (2018). Multi-objective virtual network embedding algorithm based on q-learning and curiosity-driven. *EURASIP Journal on Wireless Communications and Networking*, *2018*, 150. doi:10.1186/s13638-018-1170-x.

Herrera, J. G., & Botero, J. F. (2016). Resource allocation in NFV: A comprehensive survey. (pp. 518–532). volume 13. doi:10.1109/TNSM.2016.2598420.

Huawei, HKT and GSA (2019). *Indoor 5G Scenario Oriented White Paper*. Technical Report (Version 3.0) Huawei Technologies Co. Ltd. Shenzen, CN.

Jmila, H., & Blanc, G. (2019). Designing security-aware service requests for NFV-enabled networks. In *28th International Conference on Computer Communication and Networks, ICCCN 2019, Valencia, Spain, July 29 - August 1, 2019* (pp. 1–9). doi:10.1109/ICCCN.2019.8847058.

Khan, R., Kumar, P., Jayakody, D. N. K., & Liyanage, M. (2020). A survey on security and privacy of 5g technologies: Potential solutions, recent advancements, and future directions. *IEEE Communications Surveys Tutorials*, *22*, 196–248.

Kibalya, G., Serrat, J., Gorricho, J., Yao, H., & Zhang, P. (2019). A reinforcement learning based approach for 5G network slicing across multiple domains. doi:10.23919/CNSM46954.2019.9012674.

Kreutz, D., Ramos, F. M., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*, 14–76. doi:10.1109/JPROC.2014.2371999.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc. doi:10.1145/3065386.

Laghrissi, A., & Taleb, T. (2019). A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys Tutorials*, *21*, 1409–1434.

Li, S., Saidi, M. Y., & Chen, K. (2016). Multi-domain virtual network embedding with coordinated link mapping. In *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (pp. 1–6). doi:10.1109/SOFTCOM.2016.7772158.

Li, X., Casellas, R., Landi, G., de la Oliva, A., Costa-Perez, X., Garcia-Saavedra, A., Deiss, T., Cominardi, L., & Vilalta, R. (2017). 5G-crosshaul network slicing: Enabling multi-tenancy in mobile transport networks. *IEEE Communications Magazine*, *55*, 128–137. doi:10.1109/MCOM.2017.1600921.

Li, X., Guo, C., Gupta, L., & Jain, R. (2019a). Efficient and secure 5G core network slice provisioning based on vikor approach. *IEEE Access*, *7*, 150517–150529. doi:10.1109/ACCESS.2019.2947454.

Li, Z., Lu, Z., Deng, S., & Gao, X. (2019b). A self-adaptive virtual network embedding algorithm based on software-defined networks. *IEEE Transactions on Network and Service Management*, *16*, 362–373. doi:10.1109/TNSM.2018.2876789.

Lin, P., Wu, C., & Shih, P. (2017). Optimal placement of network security monitoring functions in nfv-enabled data centers. In *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)* (pp. 9–16). doi:10.1109/SC2.2017.10.

Liu, Y., Zhang, H., Liu, J., & Yang, Y. (2017). A new approach for delivering customized security everywhere: Security service chain. In *Security and Communication Networks* (pp. 1–17). doi:10.1155/2017/9534754.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., & Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, *18*, 236–262. doi:10.1109/COMST.2015.2477041.

Ni, Y., Huang, G., Wu, S., Li, C., Zhang, P., & Yao, H. (2019). A PSO based multi-domain virtual network embedding approach. *China Communications*, *16*, 105–119. doi:10.25046/aj020370.

Olimid, R. F., & Nencioni, G. (2020). 5g network slicing: A security overview. *IEEE Access*, *8*, 99999–100009.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., & Folgueira, J. (2017). Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges. *IEEE Communications Magazine*, *55*, 80–87. doi:10.1109/MCOM.2017.1600935.

Sciancalepore, V., Costa-Perez, X., & Banchs, A. (2019). RL-NSB: Reinforcement learning-based 5G network slice broker. *IEEE/ACM Transactions on Networking*, *27*, 1543–1557. doi:10.1109/TNET.2019.2924471.

Shahriar, N., Taeb, S., Chowdhury, S. R., Tornatore, M., Boutaba, R., Mitra, J., & Hemmati, M. (2019). Achieving a fully-flexible virtual network embedding in elastic optical networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications* (pp. 1756–1764). doi:10.1109/INFOCOM.2019.8737601.

Shameli-Sendi, A., Jarraya, Y., Pourzandi, M., & Cheriet, M. (2017). Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Transactions on Services Computing*, . doi:10.1109/TSC.2016.2616867.

Song, A., Chen, W.-N., Gu, T., Yuan, H., Kwong, S., & Zhang, J. (2019). Distributed virtual network embedding system with historical archives and set-based particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, . doi:10.1109/TSMC.2018.2884523.

Soualah, O., Mechtri, M., Ghribi, C., & Zeghlache, D. (2019). Online and batch algorithms for VNFs placement and chaining. *Computer Networks*, *158*, 98–113. doi:10.1016/j.comnet.2019.01.041.

Su, R., Zhang, D., Venkatesan, R., Gong, Z., Li, C., Ding, F., Jiang, F., & Zhu, Z. (2019). Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models. *IEEE Network*, *33*, 172–179. doi:10.1109/MNET.2019.1900024.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA; London: MIT Press.

Xie, Y., Liu, Z., Wang, S., & Wang, Y. (2016). Service function chaining resource allocation: A survey. volume abs/1608.00095.

Xing, C., Lan, J., & Hu, Y. (2013). Virtual network with security guarantee embedding algorithms. *Journal of Computers*, *8*, 2782–2789. doi:10.4304/jcp.8.11.2782-2788.

Yuan, Y., Wang, C., Peng, S., & Sood, K. (2019). Topology-oriented virtual network embedding approach for data centers. *IEEE Access*, *7*, 2429–2438. doi:10.1109/ACCESS.2018.2886270.

Zegura, E. W., Calvert, K. L., & Bhattacharjee, S. (1996). How to model an internetwork. In *Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies Conference on The Conference on Computer Communications - Volume 2* INFOCOM'96 (p. 594–602). USA: IEEE Computer Society. doi:10.1109/INFCOM.1996.493353.

Zhang, P., Li, H., Ni, Y., Gong, F., Li, M., & Wang, F. (2020). Security aware virtual network embedding algorithm using information entropy TOPSIS. *Journal of Network and Systems Management*, *28*, 35–57. doi:10.1007/s10922-019-09500-4.

Zhang, P., Yao, H., Qiu, C., & Liu, Y. (2018). Virtual network embedding using node multiple metrics based on simplified electre method. *IEEE Access*, *6*, 37314–37327. doi:10.1109/ACCESS.2018.2847910.

appendices

A. Notations

Notation	Description
	Substrate network notations
G_s	substrate network
$n_s \in N_s$	set of substrate nodes
$l_s \in L_s$	set of substrate links
$c(n_s)$	available capacity of substrate node n_s
$bw(l_s)$	available bandwidth on substrate link l_s
	Slice request notations
G_r	slice request
$n_r, m_r \in N_r$	set of virtual network functions
$l_r \in L_r$	set of virtual links
$\mathfrak{N}_r$	set of VNFs in all slice requests
$\mathfrak{L}_r$	set of virtual links in all slice requests.
$c(n_r)$,	required capacity of VNF n_r
$bw(l_r)$	bandwidth required by the virtual link l_r
	Security constraints notations
$MC(n_r)$	VNFs that must be co-located with n_r
$MC(l_r)$	virtual links that must be wo-located with l_r
$FC(n_r)$	VNFs that must NOT be co-located with n_r
$FC(l_r)$	virtual links that must NOT be co-located with l_r
$MH(n_r)$	physical nodes allowed to host the VNF n_r
$MH(l_r)$	physical links allowed to host l_r
	Evaluation metrics notation
$\mathbb{A}$	rate of successfully accepted service graphs
$\mathbb{RC}$	average revenue to cost ratio for accepted requests
$\mathbb{T}$	execution time
$\mathbb{ns}$	average node stress
$\mathbb{ls}$	average link stress
$\mathcal{AR}$	set of accepted requests
$\mathbb{R}(G_r)$	request revenue for slice G_r
$\mathbb{C}(G_r)$	embedding cost of slice G_r
$\mathcal{N}$	The number of virtual components having non empty security set for a given constraint type
$\mathcal{S}$	The size of security set associated to a virtual component of the set $\mathcal{N}$

Table 7: Notations used throughout the article.