Addad, Rami; Bagaa, Miloud; Taleb, Tarik; Cadette Dutra, Diego Leonel; Flinck, Hannu

Optimization model for Cross-Domain Network Slices in 5G Networks

Published in:
IEEE Transactions on Mobile Computing

DOI:
10.1109/TMC.2019.2905599

Published: 01/05/2020

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Addad, R., Bagaa, M., Taleb, T., Cadette Dutra, D. L., & Flinck, H. (2020). Optimization model for Cross-Domain Network Slices in 5G Networks. *IEEE Transactions on Mobile Computing*, *19*(5), 1156-1169. Article 8668438. https://doi.org/10.1109/TMC.2019.2905599

Optimization model for Cross-Domain Network Slices in 5G Networks

Rami Akrem Addad[1], Miloud Bagaa[1], Tarik Taleb[1,2,5], Diego Leonel Cadette Dutra[3] and Hannu Flinck[4]
[1] Aalto University, Espoo, Finland, [2] Oulu University, Finland
[3] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[4] Nokia Bell Labs, Espoo, Finland, [5] Sejong University, Seoul, Korea

***Abstract*—Network Slicing (NS) is a key enabler of the upcoming 5G system and beyond, leveraging on both Network Function Virtualization (NFV) and Software Defined Networking (SDN), NS will enable a flexible deployment of Network Functions (NFs) belonging to multiple Service Function Chains (SFC) over various administrative and technological domains. Our novel architecture addresses the complexities and heterogeneities of verticals targeted by 5G systems, whereby each slice consists of a set of SFCs, and each SFCs handling specified traffic within the slice. In this paper, we propose and evaluate a MILP optimization model to solve the complexities that arise from this new environment, our proposed model enables a cost-optimal deployment of network slices allowing a mobile network operator to efficiently allocate the underlying layer resources according to its users' requirements. We also design a greedy-based heuristic to investigate the possible trade-offs between execution runtime and network slice deployment. For each network slice, the proposed solution guarantees the required delay and the bandwidth, while efficiently handling the use of both the VNF nodes and the physical nodes, reducing the service provider Operating Expenditure (OPEX).**

***Index Terms*—5G, Next Generation system (NextGen), Network slicing, Service Function Chaining, Network Functions, Network Softwarization, Optimization.**

I. Introduction

5G systems unlike the previous generation of mobile networks are expected to rely on both the advancement of physical infrastructures represented by the introduction of Millimeter waves, massive MIMO, full duplex, beamforming, and small cells; as well as the emergence of SDN and NFV [2]. By introducing the logical infrastructure abstraction, the 5G mobile networks will revamp modern network infrastructures using SDN and NFV as key enabler technologies towards softwarized networks. Network Softwarization is the core concept supporting the 5G's use cases, i.e., enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (uRLLC), and massive Machine Type Communication (mMTC) [3], reducing both the Capital Expenditures (CAPEX) and the OPEX of the service provider, while keeping the deployment schema simple. Network Softwarization can enable high-performance improvements by offering the flexibility and modularity that are required to create multiple overlying networks. These softwarized networks' mechanisms give place to a new concept dubbed Network Slicing [4], [5].

Meanwhile, the Third Generation Partnership Project (3GPP) in its Releases 15 and 16 introduced a service-oriented 5G core network (5GCN) that entirely relies on NFs [6], [7], which increases the need for autonomous mechanisms to deploy and manage NS through operating of multiple Service Function Chains (SFC) that will dynamically steer the network traffic and flows across multiple logical and physical infrastructures [8]. For instance, a given user has a network slice that consists of two SFCs:

- The first one is used to handle the control plane part by steering the traffic through the Access and Mobility Management Function (AMF) and the Session Management Function (SMF) which are equivalent to the MME, P-Gateway Control plane (P-GWCP), and S-Gateway Control plane (S-GWCP) in the 4G system after the control and user plane separation of EPC nodes (CUPS).
- The second SFC will ensure the reliability of the data plane by steering the data flows from the AMF to the Data Network (DN) passing by the User plane Function (UPF) which represents the p-gateway User plane (P-GWUP) and the s-gateway User plane (S-GWUP) in the CUPS architecture [9], [10].

As the standards development organizations (SDOs), i.e., the Next Generation Mobile Network Alliance (NGMN), 3GPP, and International Telecommunication Union – Telecommunication Standardization Sector – (ITU-T), are instantiating network slices that contain one or more SFCs, each SFC composed by a set of NFs running inside either a logical node or a physical node. To enable this emerging approach, many NFs may require being traversed in a certain strict order, leveraging on the flexibility of NFV, Mobile Network Operators (MNOs) can deploy any particular slice type honoring its real-time requirements. However, this flexible management can lead to a huge number of active nodes in the network infrastructure that are scarcely used which leads to an inefficient network slicing deployment. Based on these observations, the contributions of this paper are:

- The introduction of a new architecture in compliance with the ETSI-NFV model and the 3GPP specifications to create a fine-grained NS;
- The formulation of a Mixed Integer Linear Programming (MILP) to achieve an efficient cross-domain network

An abridged version of this paper has been published in the proceedings of the 2018 edition of the IEEE GLOBECOM [1].

slicing deployment without having to carry about the underlying topologies (both the VNF layer and the physical layer) while satisfying all the constraints and the specifications requested by the end-user or a given vertical's application;
- The design and evaluation of a heuristic algorithm to overcome the exponential runtime and allow a quick decision-making capability.

The remainder of this paper is organized as follows. Section II summarizes the fundamental background topics and related research works. Section III describes the proposed architecture and our network model. Section IV illustrates the problem formulation and describes our proposed framework solution. Section V introduces the proposed heuristic for the reduction of the exponential runtime. Section VI presents the performance evaluation and our results analysis. Finally, Section VII concludes the paper.

II. Related work

Moens and Turck [11] presented and solved a VNF placement problem, their proposed algorithm was 16 times faster than previous solutions for a small service provider. Their solution uses a hybrid scenario where part of the services may be provided by dedicated physical hardware and the remaining part is provided using virtualized service instances. By introducing the concept of a hybrid solution, the authors consider the difference between service requests and VM requests, as services must be managed by the service provider and may be deployed either on dedicated hardware or on shared service instances while requested VMs are taken by the users who requested the service chain. However, by the introduction of service function chaining, we will be considering the VNF placement problem more related to the end-to-end communication rather than an embedding problem. Added to the fact that if we consider the network slicing concept, the sharing of a virtual instance will be more restricted to allow the desired QoE and QoS. The authors of [12] tackle a trending subject of the VNFs placement problem inside a modern optical data center. To minimize the expensive optical/electrical/optical (O/E/O) conversions between the optical steering domain and the packet domain within the same domain, the authors tried to find an optimal vNF placement for vNF chaining in packet/optical DCs under the constraints of the number of conversions to be minimized. They formulated a binary integer programming (BIP) problem and proposed an alternative efficient heuristic algorithm to solve it. As explained before, this problem is more related to optical data centers rather than common network infrastructures.

Ko et al. [13] present an optimal placement of network functions in an SFC context, the problem was solved by considering the latency required to place the service function in a given SFC. In this work, the authors create an abstraction of all the underlying network equipment by evoking only the notion of service node (SN), such an abstraction allow high-level modeling of the system. They formulated the model as an Integer Non-Linear Programming (INLP) problem based on the latency requirements, however, under certain conditions, the delay constraints are not enough to fulfill all the needs for having a reliable SFC orchestration system; more constraints need to be considered to strength their proposal. Song et al. [14] treat the problem of obtaining an optimal placement of network functions in the operators' networks. They formulated an Integer Linear Programming (ILP) problem to demonstrate the trade-off between the network cost and the computing resources cost. The network cost is mainly the bandwidth of the links between the network functions and the resources cost are the CPUs consumed by these network functions. They proved the NP-hardness of the formulated problem and provided a solution based on Hidden Markov Model (HMM). Their solution shows a reduction in the resources consumption as well as in the communication resources cost. Considering the network cost based only on the bandwidth could guarantee the satisfaction of all the desired flows. However, considering the service requests of 5G's critical use cases as URLLC, it is mandatory to also consider latency.

Jiao et al. [15] tried to maximize the traffic throughput under the constraint of the end-to-end latency in a given service function chain to obtain an optimal placement. After formulating the problem as an Integer Linear Programming (ILP), the authors use dynamic programming to solve the resulted NP-hard problem. However, the authors do not consider either the presence of the network slicing paradigm nor the satisfaction of the bandwidth requirements. In addition, the authors consider a static allocation of SFC requests rather than an elastic one which could be quite common in a softwarization world. In [16], the authors evoke the problem of coordinated NFV Resource Allocation (NFV-RA), they give a whole overview of all the past related work. The authors present an optimal solution to this issue using CPLEX and leveraging on Homogeneous Link Mapping (HLM) modeling. The authors try to minimize an overall cost which is constituted of the link cost (bandwidth, latency), the CAPEX cost and the OPEX cost, then they transform the formulation in order to present it to the optimizer (CPLEX in this case) by using the developed HLM. By trying to solve the problem of NFV-RA, the authors solve some issues related to the SFC domain, however, they do not consider the inclusion of the network slicing paradigm, which will play an important role in the future of networking and which will bring more restrictions concerning resource allocation.

Dietrich et al. [17] presented a holistic solution to the multi-provider network service embedding (NSE) problem to allow SFC mapping across multiple domains. They explained that the traffic scaling and NF location dependencies are the main challenging aspects of the problem. By leveraging linear programming formulations they derived near-optimal solutions for the NSE problem. The proposed approach is a centralized method that exposes only the important substrate network information of each domain to a third party. However, the authors did not consider the end to end communication delay as well as the notion of network slicing which creates a more sophisticated SFC deployment across multiple domains and in the same time introduce more challenging issues related to the resource sharing and communication requirements.

The authors of [18], proposed a Network Function Consolidation (NFC) modeling, then followed by an Integer Linear Programming (ILP) formulation, that they solved using a greedy-based heuristic solution. They consider a typical three-layer architecture in which they introduce the notion of application layer as well as the VNF layer and the underlying network layer. In a more understandable view, the authors tried to minimize the number of VNFs deployed in the network by allowing several network functions to be hosted in a limited number of VNFs. The proposed approach shows efficient results, however, with the entrance of the network slicing paradigm, such an approach should be reconsidered because of the sharing limit that has for objective to satisfy the end-users. In addition, the authors assume that all the components of the underlying infrastructure are similar, which will reduce the users' preferences in terms of authorized underlying instances.

Bari et al. [19] emphasize on the importance of using NFV paradigm to efficiently reduce the cost and maximize the profit for the service owner, and they showed the necessity of SFC orchestration to handle the traffic steering through different VNFs. The authors formulated an Integer Linear Programming (ILP), they solved it using CPLEX optimizer and extended it to nested bin packing SFC orchestration problems to address the exponential time obtained while maintaining a near-optimal performance. However, the authors consider the requests' routing issue while our work focuses more on the constraints related to the network resources limitations (latency, bandwidth), in addition to the absence of network slicing notion which may introduce more difficulties during the modeling phase, as well as problem-solving.

With respect to the cited works, in this study, we introduce a cost-optimal deployment of network slices by considering the embedding constraints, end-to-end communication delays and bandwidth limitations in network slicing environments. Seeing that new use-cases rely entirely on NS, SFC, and NFs to enable a highly mobile environment, this work is a must for achieving a fast network slices deployment for the upcoming 5G mobile systems and beyond, while conserving latency, bandwidth and resources access requirements.

III. Proposed Architecture & Network model

Fig.1 depicts the main overview architecture suggested in this paper, we have divided the architecture into three layers, as it integrates the ETSI-NFV model and the 3GPP entities to enable the monitoring, selection and creation process of the virtual instances. The physical layer consists of a set of servers and routers; in this layer, the servers are grouped into a set of data centers that communicate between themselves through the physical network. A set of routers would be used as connectors for connecting different data centers. In the NFV model, this layer refers to the NFV infrastructure (NFVI) and would be controlled by the Virtualized Infrastructure Manager (VIM) presented in the same figure. The VNF layer consists of a set of virtual network functions (VNFs) created on top of the servers, each VNF being dedicated to one or many functionalities during the forwarding of different data traffics. The VNF layer is managed by the VNF Manager (VNFM) that ensures the life-cycle management of all VNF instances spreading over multiple administrative domains. The slice layer, which runs on top of the VNF layer, consists of a set of slices that are dedicated to different services, e.g., health-care and connected cars. The traffic in each slice is routed thanks to service function chaining (SFC), where each traffic in the slice would be forwarded using a predefined order. Each slice is formed by ingress, egress nodes, as well as a set of intermediate nodes. At the reception of different packets at the ingress node, which is also called classifier, the SFC of those packets would be identified, and then the traffic would be forwarded according to that specified SFC. It is noticed that the AMF is considered the classifier in the 3GPP standardization as it is the shared entity between the control plane and the data plane. For instance, in the case of connected car management that belong to the URLLC categories, a slice can be comprised of more than one SFC inside a given network infrastructure. While the first SFC could be dedicated to the monitoring and control plane information, the second SFC could be used for applying different management actions i.e the data plane. In the following work only the core network (CN) part is considered, while the Radio part was not studied, however, the proposed solution can be also used to deploy RAN slices or RAN/CN slices if the RAN part is an NF on top of the cloud/data-center.

Let $G(V, E, W, C)$ be a weighted graph that represents the physical layer. Each node represents a server in a data center. $V = \mathcal{H} \cup U$, where U presents the set of nodes deployed in each data center and $\mathcal{H}$ are the set of connector nodes that connect different data centers. A node in U can be either a server or a router that forwards the traffic. Meanwhile, the nodes in $\mathcal{H}$ form a wide area network (WAN) that interconnects different data centers. Each vertex $v \in V$ consists of an ordered list, whereby each element in that list describes the number of resources on that node, such as CPU, Memory, and Disk. For instance, a vertex v can be presented as follows $(CPU, RAM, Disk, I/O)$. From another side, E represents all the physical links relaying between the nodes V. Formally $(u, v) \in E$ if there is a direct link between vertices u and v. We also define $E(u, v)$ that shows the relations between two vertices $u, v \in V$. $E(u, v) = 1$ if there is a physical link between u and v, otherwise $E(u, v) = 0$. W represents the weight of every link in the physical network G in the form of another ordered list that consists of the bandwidth and the latency (Bw, L). Due to the limited capacity and the high concurrency in WAN connections (the aggregation of all data-centers' links), usually, the links' capacities between $\mathcal{H}$ are too low compared to the ones between U [20], [21]. Let $W_B(u, v)$ and $W_L(u, v)$ denote the available bandwidth and end-to-end latency between nodes u and v. Formally, $W_B(u, v) \leq E(u, v) \times \mathcal{M}$ and $W_L(u, v) \geq (1 - E(u, v)) \times \mathcal{M}$, such as $\mathcal{M}$ is a big number ($\mathcal{M} \approx +\infty$). We denote by C the characteristics of different vertices including the level of security and the IaaS that the physical node belongs to; this information could be used during the placement phase of the virtual instances, i.e., when a user or a vertical's application asks for only public IaaSs, only the nodes from the public

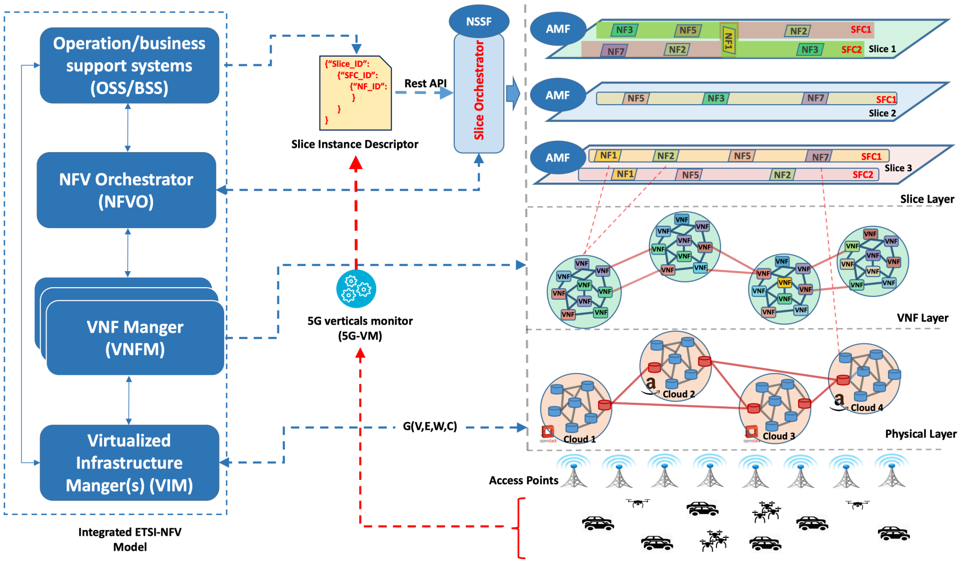


Fig. 1: Global Architecture of the proposed solution

IaaSs could be selected. For each node $v \in \mathcal{U}$, we denote by $v(C_{\mathcal{S}})$ and $v(C_{\mathcal{I}})$ the security level and the IaaS that the node v is part of. The proposed solution can easily consider more complicated characteristics by updating C.

We denote by $\mathcal{R}(\mathcal{X}, \mathcal{N}, \mathcal{P}, \mathcal{Q})$ the graph that represents the VNFs (instances of virtualization) running on top of physical nodes represented by $G(V, E, W, C)$. $\mathcal{X}$ represents the set of already existing VNFs or the ones that should be created in the network. A VNF $x \in \mathcal{X}$ is also presented by an ordered list, where each element shows the resources used by that VNF. For instance, a vertex $x \in \mathcal{X}$ can be also presented as follows $(CPU, RAM, Disk, I/O)$, where each element in the list shows the number of resources used by that VNF. While $\mathcal{N}$ presents the logical communication links between different VNFs, $\mathcal{P}$ represents the characteristics of different links. In fact, $p \in \mathcal{P}$ is also an ordered list, where each element represents the amount of resource can be offered by that link. For the sake of simplicity, we also assume that $p \in \mathcal{P}$ consists of two elements, which are the bandwidth and latency (Bw, L). We denote by $\mathcal{P}_B(u, v)$ and $\mathcal{P}_L(u, v)$ the bandwidth and end-to-end latency guaranteed between VNFs u and v, respectively. Similarly to $\mathcal{C}$, $\mathcal{Q}$ represents all sort of characteristics related to the VNF layer such as the security level, placement admission or even different complicated options not considered in this paper.

In the proposed architecture, it is assumed that a global service has multiple users that can be grouped over a set of slices where each slice is an instance of the service chain with its own owner. Moreover, the grouping method may be application specific, i.e., for autonomous driving assistance, it may be locality based. Under these considerations, it is defined that a slice (an instance of service chain) is created on the behalf of a group of users. The 5G verticals monitor (**5G-VM**) will be in charge of gathering information about the services running at different users and devices, such as the amount of bandwidth and end-to-end delay. **5G-VM** periodically monitors and detects the changes occurring at the network, including users' and devices' demands and/or their mobility, which can affect the service level agreement (SLA), then it will trigger the Slice Orchestrator (**SO**) for creating and/or rescheduling the different slices, as well as transfer the different monitoring information to the **SO** as a Slice Instance Descriptor (**SID**). By leveraging the **5G-VM**, end users and/or verticals' applications submit as a **SID** all the set of necessary specifications for the creation of a slice $s \in \mathcal{S}$ to the **SO**. By using the Network Slice Selection Function (**NSSF**) [22], which is a 3GPP functional component responsible for selecting the appropriate Network Slice instance based on the information transferred by the **SID**; and both the VNF Orchestrator (VNFO) and Operation/Business Support Systems (OSS/BSS) from the ETSI-NFV model, the **SO** will create a dynamic network slice based on chained NFs running on top of VNFs. On one side, the specifications obtained from the **SID** contain information on the service function chain $f_i \in F_s$, with $s \in \mathcal{S}$, needed for the establishment of a network slice, the ingress node (AMF if following the 3GPP standardization) and the egress node. While the ingress node classifies the incoming packets based on the pre-defined network policy traffic for the available set of SFCs F_s in the slice $s \in \mathcal{S}$, the egress node forwards the processed packets to the outside of the SFC domain (an output node).

On another side, the SFCs are composed of a set of NFs connected through virtual links. Each NF requires a certain CPU, RAM and a set of authorized nodes y_j for the deployment of a given NF. It is noticed that end-users and/or verticals' applications can impose certain affinity constraints in order to deploy their NFs, for instance, users can ask for only the public IaaS which means that only the data centers responsible for hosting the public IaaS will be taken into consideration. Depending on service characteristics, SFCs have different bandwidth and latency requirements. In addition to the bandwidth and the latency between each two NFs, the

SFCs have the global bandwidth and latency requirements denoted by l_f and w_f that must be satisfied.

After specifying the total requirements for the slice creation, the **5G-VM** transfers the requests to the **SO** in a **SID** format, then the **SO** will take as an additional input, the number of available resources in the physical layer (transmitted from the NFVI through the NFVO). The **SO** will verify for each request from the **5G-VM** (a request can be for either one or many users), the requirements for the creation and finally leverage the **NSSF** function to select the right Network Slice, i.e., a creation model driven from the compliance of our proposed architecture with the 3GPP entities and the ETSI-NFV model [23]. For instance, let's assume that the Policy Control Function (PCF) which is responsible for the billing system in the 5G core network is used, it requires 5 Gb of vRAM, 3 vCPU and a set of authorized IaaS $y\{2,5,7\}$, the **SO** will use the available resources from the physical layer to place the NF in the right place in concordance with the other NFs that belong to the same SFC in order to satisfy also the requirements of the connectivity (bandwidth and latency) and finally create the desired slice.

IV. Cost aware network slice management for enabling 5G verticals

Based on the observation that one or more NFs should run on top of a VNF, it is obvious that the number of VNFs does not exceed the number of NFs in the network. As mentioned before, each slice $s \in \mathcal{S}$ has a set of SFCs F_s. Each SFC $f_i \in F_s$ consists of a set of NFs that are connected one with each other, whereby each NF has one predecessor and one successor except the first and the last NFs. While the first NF has only one successor, which is the second NF in the SFC, the last one has only one predecessor. We denote by Ψ_i^s the set of NFs in the SFC f_i. We also denote by $\Psi_{i,j}^s$ the j^{th} NF in SFC $f_i \in F_s$ at the slice $s \in \mathcal{S}$. Let we denote by Γ the number of NFs in the network. Formally, Γ can be defined as follows:

$$\Gamma = \sum_{\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s} 1 \tag{1}$$

Based on what has been discussed, we can assume that we have Γ boolean decision variables that show if the VNFs should be deployed or not. We define the following variables:

$$\forall i \in \{1...\Gamma\} : \rho_i = \begin{cases} 1 & \text{if the VNF } i \text{ should be created} \\ 0 & \text{Otherwise} \end{cases}$$

We also define the following variables:
$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, \forall k \in \{1...\Gamma\}$:

$$\mathcal{Y}_{s,i,j,k} = \begin{cases} 1 & \text{if the NF } j \text{ is running on top of VNF } k \\ 0 & \text{Otherwise} \end{cases}$$

In the Objective Function 2, we aim to minimize the number of VNFs hosting the NFs that constitute different network slices.

$$\min \sum_{v \in \{1...\Gamma\}} \rho_v \tag{2}$$

Meanwhile, the constraints will be divided into five parts: the placement constraints, the resources constraints, the links arrangements constraints, the latency aware constraints, and the bandwidth aware constraints. Each part consists of the Slice-VNF layer mapping and the VNF-Physical layer mapping.

A. Placement Constraints

In this subsection constraints related to the three layer placement will be introduced. We start by the Slice-VNF layer mapping in constraint 3 and 4, then we continue with the VNF-Physical layer mapping in constraint 5 and 6. Finally, we deduce a three layer mapping starting from constraint 7 to constraint 11.

Constraint 3 ensures that each NF should run on top of only one VNF;
$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s$:

$$\sum_{k \in \{1...\Gamma\}} \mathcal{Y}_{s,i,j,k} = 1 \tag{3}$$

Constraint 4 shows that if a given NF is running on top of a VNF, this VNF must be created;
$\forall v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s$:

$$\rho_v \geq \mathcal{Y}_{s,i,j,v} \tag{4}$$

We also define the following variables in order to introduce the VNF-physical mapping:
$\forall u \in \{1...\Gamma\}, \forall v \in V$:

$$\mathcal{X}_{u,v} = \begin{cases} 1 & \text{if the VNF u is running on top of the bare metal server v.} \\ 0 & \text{Otherwise} \end{cases}$$

The following constraint ensures that each VNF runs on top of one bare metal server at most. If it is not running on top of any server, this means that the VNF is not instantiated.
$\forall u \in \{1...\Gamma\}$:

$$\sum_{\forall v \in V} \mathcal{X}_{u,v} \leq \rho_u \tag{5}$$

We denote by y_j the list of authorized nodes of the NF j. It is noticed that the y_j is mainly used to enable a fine-grained placement for the proposed solution. The constraint 6 ensures that if a VNF is not instantiated, then it should not hold a NF.
$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, \forall k \in \{1...\Gamma\}$:

$$\mathcal{Y}_{s,i,j,k} \leq \sum_{v \in y_j} \mathcal{X}_{k,v} \tag{6}$$

Constraint 7 ensures that the VNF is instantiated in authorized bare metal servers.
$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, k \in \{1...\Gamma\}$:

$$\sum_{v \in y_j} \mathcal{Y}_{s,i,j,k} \times \mathcal{X}_{k,v} = 1 \tag{7}$$

However equation (7) is not linear. In order to make the optimization linear, we add the following constraints and variables:

Firstly, we add the following boolean decision variables:

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, \forall k \in \{1...\Gamma\}, \forall v \in y_j$: $\mathcal{Y}^*_{s,i,j,k,v}$

The constraint 8 guarantees that each NF is mapped to only one VNF which will be in its turn mapped to only one bare metal node.

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, k \in \{1...\Gamma\}$:

$$\sum_{v \in y_j} \mathcal{Y}^*_{s,i,j,k,v} = 1 \tag{8}$$

Then, we add the constraints 9, 10 and 11 to consolidate our transformation. Formally, $\mathcal{Y}^*_{s,i,j,k,v} = \mathcal{Y}_{s,i,j,k}$ if $\mathcal{X}_{k,v} = 1$, otherwise $\mathcal{Y}^*_{s,i,j,k,v} = 0$:

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, \forall k \in \{1...\Gamma\}, \forall v \in y_j$:

$$\mathcal{Y}^*_{s,i,j,k,v} \leq \mathcal{Y}_{s,i,j,k} \tag{9}$$

$$\mathcal{Y}^*_{s,i,j,k,v} \leq \mathcal{X}_{k,v} \tag{10}$$

$$\mathcal{Y}^*_{s,i,j,k,v} \geq \mathcal{X}_{k,v} + \mathcal{Y}_{s,i,j,k} - 1 \tag{11}$$

B. Resources Constraints

We denote by $\Re$ the set of resource type, such as CPU, RAM, Storage and so on. From the Slice Descriptor presented in Fig. 1, we can get the required resources for each NF.

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \Psi_i^s, \forall r \in \Re$:

$\mathcal{R}_{s,i,j}(r)$ denotes the required resource r of NF j.

We denote by π_v^r which is a real variable, the number of resources $r \in \Re$ used by VNF $v \in \{1...\Gamma\}$.

Constraint 12 ensures that the amount of resources in the Slice-VNF mapping layer is respected. Each NF request should not exceed the available resources in any given VNF deployed to serve network slices. $v \in \{1...\Gamma\}, \forall r \in \Re$:

$$\sum_{s \in \mathcal{S}, i \in F_s, j \in \Psi_i^s} \mathcal{R}_{s,i,j}(r) \times \mathcal{Y}_{s,i,j,v} \leq \pi_v^r \tag{12}$$

We denote by Π_u^r the amount of resources $r \in \Re$ of a node $u \in V$.

The constraint 13 guarantees that the VNF-physical layer mapping is respected. Any VNF should be instantiated in a physical node that has enough resources.

$\forall u \in V, \forall r \in \Re$:

$$\sum_{v \in \{1...\Gamma\}} \pi_v^r \times \mathcal{X}_{v,u} \leq \Pi_u^r \tag{13}$$

However, the constraint (13) is not linear. In order to translate the optimization to linear programming, we update the constraint (13) as follows:

We define the following variable $\pi^{*r}_{v,u}$ for $u \in V$, $v \in \{1...\Gamma\}$ and $r \in \Re$. Formally, $\pi^{*r}_{v,u} = \pi_v^r$ if $\mathcal{X}_{u,v} = 1$, otherwise $\pi^{*r}_{v,u} = 0$. In order to ensure that the optimization problem is linear, we add the constraints 14, 15, 16 and 17, with $\mathcal{M}$ representing a big number ($\mathcal{M} \approx \infty$).:

$\forall u \in V, \forall r \in \Re$:

$$\sum_{v \in \{1...\Gamma\}} \pi^{*r}_{v,u} \leq \Pi_u^r \tag{14}$$

$\forall u \in V, v \in \{1...\Gamma\}, \forall r \in \Re$:

$$\pi^{*r}_{v,u} \leq \pi_v^r + \mathcal{M} \times (1 - \mathcal{X}_{v,u}) \tag{15}$$

$$\pi_v^r \leq \pi^{*r}_{v,u} + \mathcal{M} \times (1 - \mathcal{X}_{v,u}) \tag{16}$$

$$\pi^{*r}_{v,u} \leq \mathcal{M} \times \mathcal{X}_{v,u}, \tag{17}$$

C. Links Arrangement Constraints

The subsection IV-C introduces all the variables and constraints in relation to the links arrangement of different levels of nodes.

From constraint 18 to 21, we define the variables that have a relationship with the Slice-VNF layer. In the following, we assume that a VNF can hosts multiple NFs.

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi_i^s - \Psi_{i,1}^s\}, \forall a \in \{1...\Gamma\}, \forall b \in \{1...\Gamma\}$:

$$\mathcal{Z}^{j-1,j}_{s,i,a,b} = \begin{cases} 1 & \text{if the traffic between j and j-1} \\ & \text{pass through the link (a,b)} \\ 0 & \text{Otherwise} \end{cases}$$

We have the constraint 18 that ensures the presence of a link between each two consecutive NFs.

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi_i^s - \Psi_{i,1}^s\}$:

$$\sum_{a \in \{1...\Gamma\}} \sum_{b \in \{1...\Gamma\}} \mathcal{Z}^{j-1,j}_{s,i,a,b} = 1 \tag{18}$$

We have also the following inequalities in 19, 20 and 21 which guarantee that if there is a link between NF$_{j-1}$ and NF$_j$, the VNF a and b are hosting respectively NF$_{j-1}$ and NF$_j$ and deployed in the network:

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi_i^s - \Psi_{i,1}^s\}, \forall a \in \{1...\Gamma\}, \forall b \in \{1...\Gamma\}$:

$$\mathcal{Z}^{j-1,j}_{s,i,a,b} \leq \mathcal{Y}_{s,i,j-1,a} \tag{19}$$

$$\mathcal{Z}^{j-1,j}_{s,i,a,b} \leq \mathcal{Y}_{s,i,j,b} \tag{20}$$

$$\mathcal{Z}^{j-1,j}_{s,i,a,b} \geq \mathcal{Y}_{s,i,j-1,a} + \mathcal{Y}_{s,i,j,b} - 1 \tag{21}$$

$\forall u, v \in \{1...\Gamma\}, \forall (a,b) \in E$:

$$\xi^{a,b}_{u,v} = \begin{cases} 1 & \text{if the communication between u} \\ & \text{and v uses the link (a,b).} \\ 0 & \text{Otherwise} \end{cases}$$

Constraints 22, 23 and 24 need to be introduced to guarantee the VNF-physical layer mapping. They ensure that if there is a link between VNF$_a$ and VNF$_b$, the physical nodes u and v are hosting respectively VNF$_a$ and VNF$_b$ and deployed in the network:

$\forall u, v \in \{1...\Gamma\}, \forall (a,b) \in E$:

$$\xi^{a,b}_{u,v} \leq \mathcal{X}_{u,a} \tag{22}$$

$$\xi^{a,b}_{u,v} \leq \mathcal{X}_{v,b} \tag{23}$$

$$\xi^{a,b}_{u,v} \geq \mathcal{X}_{u,a} + \mathcal{X}_{v,b} - 1 \tag{24}$$

D. Latency Aware Constraints

The constraints in IV-D guarantee that the links have the requested end-to-end latency for ensuring a good system functionality.

Starting from constraint 25 to 30, the latency of the Slice-VNF layer mapping is ensured. We define the following variables:

$\phi^L_{j-1,j}$ a real variable that shows the maximum delay between the NF$_{j-1}$ and NF$_j$ in SFC i at the slice s.

The constraint 25 ensures that the desired end-to-end latency is maintained in the slice layer.

$\forall s \in \mathcal{S}, \forall i \in F_s$:

$$\sum_{j \in \{\Psi^s_i - \Psi^s_{i,1}\}} \phi^L_{j-1,j} \leq l^{fi} \tag{25}$$

We also define the following variables:

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}$:

$\Phi^L_{u,v}$ a real variable that shows the maximum delay between the VNF u and v.

The constraint 26 ensures that if the communication between NF$_{j-1}$ and NF$_j$ uses the link u, v, then the delay between VNF$_u$ and VNF$_v$ must be smaller than the delay between NF$_{j-1}$ and NF$_j$.

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi^s_i - \Psi^s_{i,1}\}$:

$$\Phi^L_{u,v} \leq \phi^L_{j-1,j} \times \mathcal{Z}^{j-1,j}_{s,i,u,v} \tag{26}$$

However, inequality (26) is not linear. In order to make the optimization problem linear, we introduce the following variables and constraints.

Firstly, we define the following real variables:

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi^s_i - \Psi^s_{i,1}\} : \phi^{L,u,v}_{s,i,j}$

With $\mathcal{M}$ as a big number ($\mathcal{M} \approx \infty$) and $\phi^{L,u,v}_{s,i,j} = \phi^L_{j-1,j}$ if $\mathcal{Z}^{j-1,j}_{s,i,u,v} = 1$, otherwise $\phi^{L,u,v}_{s,i,j} = 0$, we add the following constraints:

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi^s_i - \Psi^s_{i,1}\}$:

$$\phi^{L,u,v}_{s,i,j} \leq \phi^L_{j-1,j} + (1 - \mathcal{Z}^{j-1,j}_{s,i,u,v}) \times \mathcal{M} \tag{27}$$

$$\phi^L_{j-1,j} \leq \phi^{L,u,v}_{s,i,j} + (1 - \mathcal{Z}^{j-1,j}_{s,i,u,v}) \times \mathcal{M}, \tag{28}$$

$$\phi^{L,u,v}_{s,i,j} \geq (1 - \mathcal{Z}^{j-1,j}_{s,i,u,v}) \times \mathcal{M}, \tag{29}$$

$$\Phi^L_{u,v} \leq \phi^{L,u,v}_{s,i,j} \tag{30}$$

From the constraint number 31 to constraint number 36, we ensure that the delay of the VNF-physical layer mapping is taken into consideration.

Firstly, we define the following real variable:

$\forall (a,b) \in E : \Upsilon^L_{a,b}$ a real variable that represents maximum latency between the physical node a and b.

Constraint 31 ensures that if the VNF$_u$ and the VNF$_v$ pass through the physical link between the underlying node a and b, the available latency between the physical nodes must be respected.

$\forall u, v \in \{1...\Gamma\}, \forall (a,b) \in E$:

$$\Upsilon^L_{a,b} \leq \Phi^L_{u,v} \times \xi^{a,b}_{u,v} \tag{31}$$

However, equation (31) is not linear. In order to make the optimization problem linear, we define the following variables and constraints:

Firstly, we add the following real variables:

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}, \forall (a,b) \in E\} : \Omega^L_{u,v,a,b}$.

With $\mathcal{M}$ as a big number ($\mathcal{M} \approx \infty$) and $\Omega^L_{u,v,a,b} = \Phi^L_{u,v}$ if $\xi^{a,b}_{u,v} = 1$, otherwise $\Omega^L_{u,v,a,b} = 0$, we add the following constraints:

$$\Omega^L_{u,v,a,b} \leq \Phi^L_{u,v} + (1 - \xi^{a,b}_{u,v}) \times \mathcal{M} \tag{32}$$

$$\Phi^L_{u,v} \leq \Omega^L_{u,v,a,b} + (1 - \xi^{a,b}_{u,v}) \times \mathcal{M} \tag{33}$$

$$\Omega^L_{u,v,a,b} \geq (1 - \xi^{a,b}_{u,v}) \times \mathcal{M} \tag{34}$$

$$\Upsilon^L_{a,b} \leq \Omega^L_{u,v,a,b} \tag{35}$$

The constraint 36 ensures that always the requested latency is bigger than the available one between the physical nodes.

$$\Upsilon^L_{a,b} \geq \mathcal{W}_L(a,b) \tag{36}$$

E. Bandwidth Aware Constraints

The following constraints guarantee that each link has enough bandwidth for ensuring a good system functionality.

Starting from constraint 37 to 42, the latency of the Slice-VNF layer mapping is ensured. We define the following variables:

$\delta^B_{j-1,j}$ a real variable that shows the minimum bandwidth between the NF$_{j-1}$ and NF$_j$ in SFC i at the slice s.

Constraint 37 ensures that the required bandwidth is respected between each two successive NFs.

$\forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi^s_i - \Psi^s_{i,1}\}$:

$$\delta^B_{j-1,j} \geq w^{fi} \tag{37}$$

The constraint 38 ensures that the bandwidth between VNF$_u$ and VNF$_v$ equals the sum of all the bandwidth used by different NFs.

$\Delta^B_{u,v}$ a real variable that shows the minimum bandwidth between the VNF u and v.

$\forall u, v \in \{1...\Gamma\}$:

$$\Delta^B_{u,v} \geq \sum_{s\in\mathcal{S}, i\in F_s, j\in\{\Psi_i^s - \Psi_{i,1}^s\}} \delta^B_{j-1,j} \times \mathcal{Z}^{j-1,j}_{s,i,u,v} \tag{38}$$

However, inequality (38) is not linear. In order to make the optimization problem linear, we introduce the following variables and constraints.

Firstly, we add the following real variables: $\forall u, v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi_i^s - \Psi_{i,1}^s\} : \delta^{B,u,v}_{s,i,j}$

With $\mathcal{M}$ as a big number ($\mathcal{M} \approx \infty$) and $\delta^{B,u,v}_{s,i,j} = \delta^B_{j-1,j}$ if $\mathcal{Z}^{j-1,j}_{s,i,u,v} = 1$, otherwise $\delta^{B,u,v}_{s,i,j} = 0$, we add the following constraints:

$\forall u, v \in \{1...\Gamma\}, \forall s \in \mathcal{S}, \forall i \in F_s, \forall j \in \{\Psi_i^s - \Psi_{i,1}^s\}$:

$$\delta^{B,u,v}_{s,i,j} \leq \delta^B_{j-1,j} + (1 - \mathcal{Z}^{j-1,j}_{s,i,u,v}) \times \mathcal{M} \tag{39}$$

$$\delta^B_{j-1,j} \leq \delta^{B,u,v}_{s,i,j} + (1 - \mathcal{Z}^{j-1,j}_{s,i,u,v}) \times \mathcal{M} \tag{40}$$

$$\delta^{B,u,v}_{s,i,j} \leq \mathcal{Z}^{j-1,j}_{s,i,u,v} \times \mathcal{M} \tag{41}$$

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}\}$:

$$\Delta^B_{u,v} \geq \sum_{s\in\mathcal{S}, i\in F_s, j\in\{\Psi_i^s - \Psi_{i,1}^s\}} \delta^{B,u,v}_{s,i,j} \tag{42}$$

The remaining constraints ensure that the bandwidth constraints of the VNF-physical layer mapping are taken into consideration.

Firstly, we define the following real variable:

$\forall (a, b) \in E : \Lambda^B_{a,b}$ a real variable that represents minimum bandwidth between the physical node a and b.

Then, we add the following constraints.

Constraint 43 ensures that if the VNF$_u$ and the VNF$_v$ pass through the physical link between the underlying node a and b, the available bandwidth between the physical nodes must be respected.

$\forall (a, b) \in E$:

$$\Lambda^B_{a,b} \geq \sum_{u,v\in\{1...\Gamma\}} \Delta^B_{u,v} \times \xi^{a,b}_{u,v} \tag{43}$$

However, equation (43) is not linear. In order to make the optimization problem linear, we define the following variables and constraints:

Firstly, we add the following real variables:

$\forall u \in \{1...\Gamma\}, \forall v \in \{1...\Gamma\}, \forall (a, b) \in E\} : \Theta^B_{u,v,a,b}$.

With $\mathcal{M}$ as a big number ($\mathcal{M} \approx \infty$) and $\Theta^B_{u,v,a,b} = \Delta^B_{u,v}$ if $\xi^{a,b}_{u,v} = 1$, otherwise $\Theta^B_{u,v,a,b} = 0$, we add the following constraints:

$\forall u, v \in \{1...\Gamma\}, \forall (a, b) \in E$:

$$\Theta^B_{u,v,a,b} \leq \Delta^B_{u,v} + (1 - \xi^{a,b}_{u,v}) \times \mathcal{M} \tag{44}$$

$$\Delta^B_{u,v} \leq \Theta^B_{u,v,a,b} + (1 - \xi^{a,b}_{u,v}) \times \mathcal{M} \tag{45}$$

$$\Theta^B_{u,v,a,b} \leq \xi^{a,b}_{u,v} \times \mathcal{M} \tag{46}$$

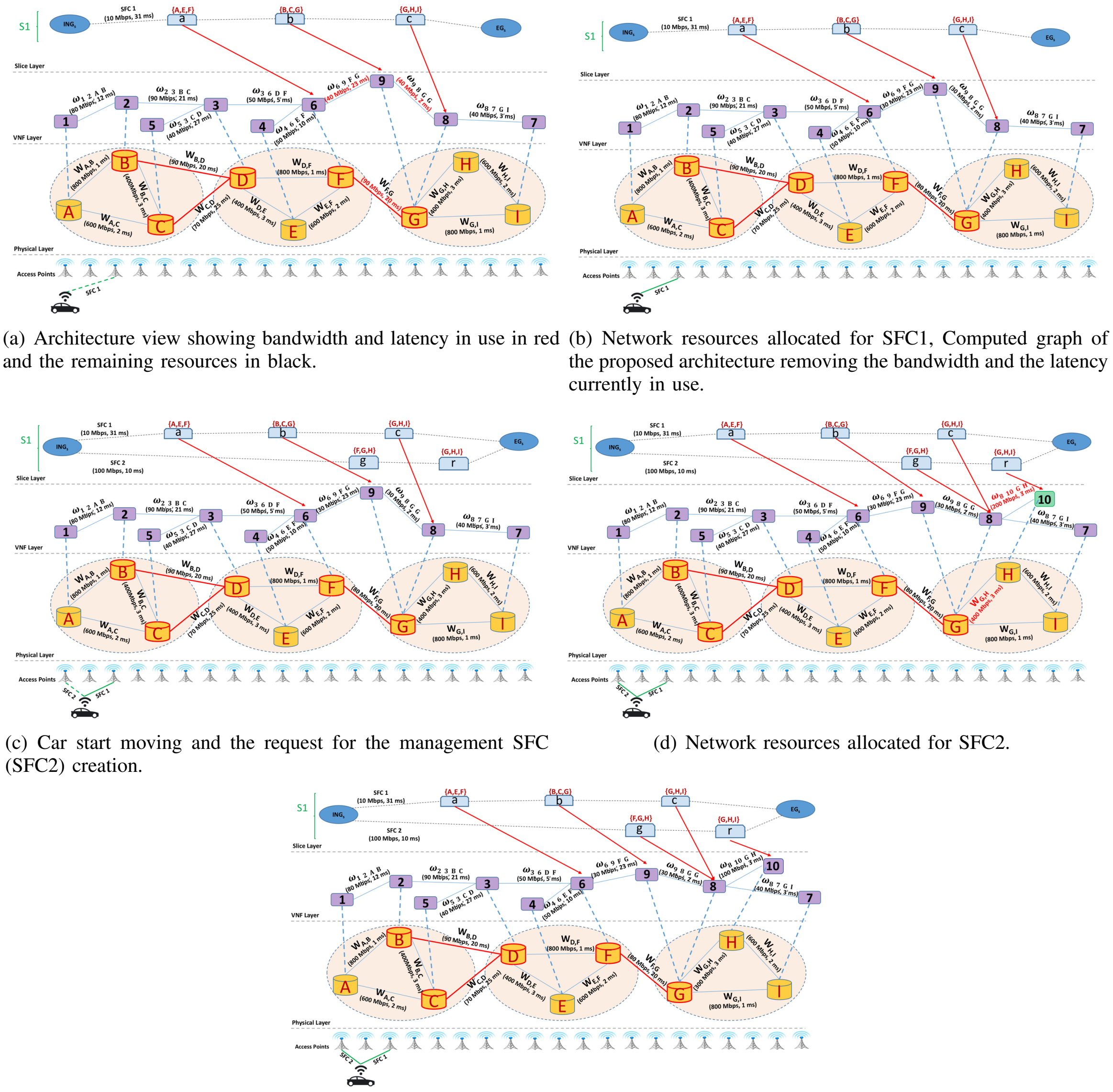
$$\Lambda^B_{a,b} \geq \sum_{u,v\in\{1...\Gamma\}} \Theta^B_{u,v,a,b} \tag{47}$$

The constraint 48 ensures that always the requested bandwidth is smaller than the available one between the physical nodes.

$$\Lambda^B_{a,b} \leq \mathcal{W}_B(a, b) \tag{48}$$

For the sake of facility, a detailed example will be introduced to illustrate the operations of our proposed solution. Fig. 2 represents a simple three-layer architecture, which consists of six data-centers named from A to I in the physical layer, a set of already deployed VNFs numbered from 1 to 9 in the VNF layer. Also note that for clarity, the bandwidth and the latency will be both represented by W and ω respectively in the VNF layer and the physical layer, we also omitted the **5G-VM**, the **SID**, the **SO** and the integrated ETSI-NFV model from Fig. 2, they are responsible for the creation of different kinds of network slices after receiving requests from end users and the selection part from the **NSSF**.

Let's assume that we have a connected car management scenario; in that case, and for safety reasons, we need at least one mandatory network slice containing an SFC dedicated for the monitoring and control plane information, by virtue of simplicity, this SFC is not shown in the Fig. 2. In Fig. 2(a) we assume that the connected car was stopped and a passenger inside attaches to the access network (AN) and requests a video streaming service that requires 10Mbps of bandwidth and 31ms of latency. By collecting the request, the **5G-VM**, the **SID**, the **NSSF** and the **SO** will coordinate and create a slice dubbed S1 in the Slice layer, S1 contains 3 NFs a, b and c deployed in VNFs 6, 9 and 7, respectively. The Fig. 2(a) shows the bandwidth and latency resources partially in use as highlighted by the red numbers between the VNFs 6, 9, and 7 in the VNF layer, and between nodes F and G in the physical layer. Based on this topology, we update our reference graphs G and $\mathcal{R}$ by removing all used resources as depicted in Fig. 2(b). In Fig. 2(c), the connected car starts moving and for safety reasons, a second SFC is needed to handle the monitoring tasks and the measurements related to the mobile connected car, therefore the connected car requests the creation of this second SFC. However, that second SFC is resource consuming and needs at least 100Mbps of bandwidth and 10ms of latency as its a delay-sensitive service (URLLC), added to that, the NF$_g$ and the NF$_r$ have some restrictions in terms of deployment in the physical layer (location constraints), for instance, NF$_r$ can only be deployed on the bare-metal servers "G", "H" and "I"; here we can clearly observe the constraints 8, 9, 10, and 11. The Fig. 2(d) shows the instantiation of a new VNF numbered as 10 to host the NF$_r$, the bandwidth, and latency resources partially in use are also highlighted using the red color between the VNF 8 and the newly created one 10 in the VNF layer, and between node G and H in the physical layer. Fig. 2(e)


S1
SFC 1 (10 Mbps, 31 ms)
SFC 2 (100 Mbps, 10 ms)
ING
EG
{A,E,F} a
{B,C,G} b
{G,H,I} c
{F,G,H} g
{G,H,I} r
Slice Layer
VNF Layer
Physical Layer
Access Points
$\omega_{1\ 2\ A\ B}$ (80 Mbps; 12 ms)
$\omega_{2\ 3\ B\ C}$ (90 Mbps; 21 ms)
$\omega_{3\ 6\ D\ F}$ (50 Mbps, 5 ms)
$\omega_{5\ 3\ C\ D}$ (40 Mbps; 27 ms)
$\omega_{4\ 6\ E\ F}$ (50 Mbps, 10 ms)
$\omega_{6\ 9\ F\ G}$ (30 Mbps; 23 ms)
$\omega_{9\ 8\ G\ G}$ (30 Mbps; 2 ms)
$\omega_{8\ 7\ G\ I}$ (40 Mbps, 3 ms)
$\omega_{8\ 10\ G\ H}$ (100 Mbps, 3 ms)
1
2
3
4
5
6
7
8
9
10
$W_{A,B}$ (800 Mbps, 1 ms)
$W_{B,D}$ (90 Mbps, 20 ms)
$W_{B,C}$ (400 Mbps, 3 ms)
$W_{A,C}$ (600 Mbps, 2 ms)
$W_{C,D}$ (70 Mbps, 25 ms)
$W_{D,F}$ (800 Mbps, 1 ms)
$W_{D,E}$ (400 Mbps, 3 ms)
$W_{E,F}$ (600 Mbps, 2 ms)
$W_{F,G}$ (80 Mbps, 30 ms)
$W_{G,H}$ (400 Mbps, 3 ms)
$W_{H,I}$ (600 Mbps, 2 ms)
$W_{G,I}$ (800 Mbps, 1 ms)
A
B
C
D
E
F
G
H
I
SFC 1
SFC 2

(a) Architecture view showing bandwidth and latency in use in red and the remaining resources in black.

(b) Network resources allocated for SFC1, Computed graph of the proposed architecture removing the bandwidth and the latency currently in use.

(c) Car start moving and the request for the management SFC (SFC2) creation.

(d) Network resources allocated for SFC2.

(e) Network resources reference graph is recalculated.

Fig. 2: Example of our proposed solution for cost optimal network slicing deployment.

represents the graphs G and $\mathcal{R}$ in the physical layer and VNF layer, respectively after the re-computation. It is noticed that all the constraints related to the link arrangement, latency and bandwidth are used in each request by our proposed solution. It is also noticed that all the instantiated NFs are supposed to represent one of the proposed NFs in the 5GCN service-based architecture, however, for reason of clarity we choose to name them using simple alphabetic identification. Finally, the obtained network slice is shared between the end-user requesting the video streaming service and the car that requires two separated SFCs for handling both the control plane information and the monitoring tasks for high-speed mobility.

F. Final model

Algorithm 1, dubbed **Optimal Cross-domain Network Slices Deployment**, summarizes these constraints. For each new request for the slice creation, the distribution of the NFs that form the slices has to be recomputed. This algorithm is triggered by either the reception of a request for creating a new slice or the update of an existing slice: whether it be the arrival of a new user (end-user or vertical's application) or the mobility of an existing one. The input parameters of the proposed solution are the graph G that represents the physical layer and the requests' specifications (authorized IaaS, bandwidth, end-to-end latency and so on). At each execution of the Algorithm, a new configuration is computed thanks to the optimization problem defined below, and then that configuration is returned to the orchestrator to enforce it. As shown in the Algorithm, the main control loop re-executes a waiting function to receive a new request either for creating a new slice or updating existing according to the changes happened at the three-layer network topology.

After introducing all the needed constraints and their respective transformations, the final model to optimize is defined as follow:

$$\min \sum_{v\in\{1...\Gamma\}} \rho_v$$
$$\text{S.t} \begin{cases} \text{Constraints } 3 - 6. \\ \text{Constraints } 8 - 12. \\ \text{Constraints } 14 - 25. \\ \text{Constraints } 27 - 30. \\ \text{Constraints } 32 - 37. \\ \text{Constraints } 39 - 42. \\ \text{Constraints } 44 - 48. \end{cases}$$

Theorem 1: Grouping NFs on VNFs and placing them on top of physical servers is NP-hard problem.

Proof The resolution introduced by the OCNSD algorithm is optimal, however, the formulated problem is NP-hard as result of the non-reasonable calculation time. One way to prove the NP-hardness of a problem is reducing it to a well-known NP-hard problem. In that case, if we simplified our problem and considered only one network slice limited to only one SFC which contains a given number of NFs, we would have a problem denoted by P where the grouping of different NFs into VNFs and placing those VNFs into physical servers is similar to the knapsack problem, which is known to be NP-hard. In the problem P, the different physical servers can be considered as knapsacks while the VNFs are considered as different objects should be filled in knapsacks. In the problem P, we aim to reduce the number of used physical servers, which is equivalent to the cost in the knapsack problem. In fact, in problem P, we aim to reduce the cost while the capacity of VNFs, in terms of CPU, memory, and disk, does not exceed the capacity of used physical servers. It's noticed that even when expunged the delay and the bandwidth the problem was NP-hard, hence adding them will increase in complexity. Thus, OCNSD is also an NP-hard problem that requires the design of a heuristic solution to obtain an engineering time of execution. □

Algorithm 1: OCNSD Algorithm.

```
Input  : G: Network Graph.
         Q: List of Requests' specification (resources,
         authorized hosts, bandwidth, end-to-end
         latency).
Output: NFs Placement,
         Allocated VNF nodes,
         Allocated DC nodes.
while True do
    read current;
    if $q\ in\ Q\ ==\ (new\ or\ updated)$ then
        $Optimization_{OCNSD}(G, Q)$;
    end
end
```

V. GDMK: Heuristic Solution

As mentioned in the previous section, achieving an optimal solution is an NP-hard problem. The optimization problem suggested in the previous solution can provide an optimal configuration for a small size network. Also, it can serve as a baseline approach for evaluating different heuristics that can be proposed in the future for providing different network configurations. In this section, we suggest a heuristic, named Greedy Distributed Multi-layer Knapsack (GDMK) solution, to provide an efficient configuration. GDMK is a greedy-based algorithm that aims to reduce the computational cost while the number of VNFs should be created in a reasonable time. Basically, the GDMK solution consists of three main ideas:

- The placement part of our network slice deployment problem can be considered as a knapsack problem, considering its three layers architecture, it can be extended to be a multi-layers knapsack problem (nested bin packing problem [24]) where the physical data-centers play the role of a knapsack for the VNF layer components and the VNF layer is considered as a knapsack hosting the SFCs with their respective NFs;
- The communication constraints (end-to-end latency and bandwidth) make our formulated problem a distributed knapsack problem where we cannot pack items (NFs in the VNF layer and VNFs in the physical layer) except if they have the required communication resources;
- By following a greedy logic when packing NFs and VNFs (NFs in the VNF layer and VNFs in the physical layer), a reasonable time will be obtained while respecting placement, latency, and bandwidth constraints. The proposed heuristic will be a greedy-based distributed multi-layer knapsack problem.

Algorithm 2 serves to describe different steps of the heuristic GDMK. As aforementioned in the previous section, the information about both physical layer and network slice layer are fed as input for the suggested solution while the information about the VNFs and the mapping about the three layers are considered as output. For this reason, in the initial step, all the required inputs related to the physical layer (data centers) and the slice layer are introduced to GDMK heuristic.

We define by γ the set of all the NFs existing in the network. Formally, γ is considered as all the NFs that should be created in each SFC F_s at each slice $s \in \mathcal{S}$ (Algorithm 2: Lines $1-8$). An NF $\nu_1 \in \gamma$ is a neighbor to another NF $\nu_2 \in \gamma$ if only ν_2 is a predecessor or a successor ν_1 in the same SFC. We denote by $\eta(\nu)$ the neighbor of ν. Let $W_L(\nu_1, \nu_2)$ and $W_B(\nu_1, \nu_2)$ denote the latency and the bandwidth between the NF ν_1 and its neighbor $\nu_2 \in \eta(\nu_1)$. As we have explained above, the set of NFs γ should run on top of VNFs δ, such that each VNF should host at least one NF. Let $\mathcal{G}$ denote the grouped set of γ, whereby each group $g \in \mathcal{G}$ consists of a set of NFs $\gamma_g \subseteq \gamma$. Let δ_g denotes the VNF that should host the group of NFs $g \in \mathcal{G}$. A VNF $g2 \in \mathcal{G}$ is a neighbor of another VNF $g1 \in \mathcal{G}$ if exist an NF $\nu_1 \in \gamma_1$ that is a neighbor to another NF $\nu_2 \in \gamma_1$. We denote by $\eta(g)$, for $g \in \mathcal{G}$, all the VNFs neighbors of the VNF δ_g.

Let $\mathcal{F}(\gamma, \mathcal{N})$ denote a function that groups the NFs γ in a set of groups $\mathcal{G}$ whose size does not exceed $\mathcal{N}$. Formally,

Algorithm 2: Greedy Distributed Multi-layer based Knapsack Heuristic (GDMK)

```
Input  :
        $G(V, E, W, C)$: A weighted graph that represents the physical layer.
        $\mathcal{S}$: A network slice.
        $F_s$: The list of network functions NFs of the slice $\mathcal{S}$.
        $l_f$: The end-to-end latency of SFC $f \in F_s$.
        $l_w$: The end-to-end bandwidth of SFC $f \in F_s$.
Output:
        $\delta$: The set of VNFs,
        $\mathcal{G}$: The grouped set of NFs,
        $\mathcal{K}$: Number of repetitions before taking the final decision.
$\gamma \leftarrow \emptyset$;
for $s \in \mathcal{S}$ do
    for $f_i \in F_s$ do
        for $\theta \in \Psi_i^s$ do
            $\gamma.append(\theta)$;
        end
    end
end
$\mathcal{N} \leftarrow |\gamma|$;
$\mathcal{R} \leftarrow \mathcal{K}$;
while True do
    $\delta_{\mathcal{T}} = \mathcal{F}(\gamma, \mathcal{N})$;
    if $\Upsilon(\delta_{\mathcal{T}}, G) == True \ \ \& \ \ |\delta_{\mathcal{T}}| < \mathcal{N}$ then
        $\delta \leftarrow \delta_{\mathcal{T}}$;
        $\mathcal{N} \leftarrow |\delta|$;
        $\mathcal{R} \leftarrow \mathcal{K}$;
    end
    else
        $\mathcal{R} \leftarrow \mathcal{R} - 1$;
        if $(\mathcal{R} == 0) \ \ \& \ \ \delta \neq \emptyset$ then
            $Break()$;
        end
    end
end
```

$\mathcal{F}(\gamma, \mathcal{N})$ returns the set of VNFs δ that should be deployed on top of V. The function $\mathcal{F}(\gamma, \mathcal{N})$ should return the list of returned VNF δ that should satisfy the following conditions:

- $\forall g1, g2 \in \mathcal{G}$, $\gamma_{g1} \cap \gamma_{g2} = \emptyset$. This means that one NF should not be deployed on two different VNFs;
- $\bigcup\limits_{g \in \mathcal{G}} \gamma_g = \gamma$. This means that all the NFs should be deployed on top of the VNFs;
- $\forall s \in \mathcal{S}, f_i \in F_s : \sum_{\nu 1 \in f_i, \nu 2 \in \eta(\nu 1)} W_L(\nu_1, \nu_2) \leq l_i^f$. This condition ensures that the sum of the hop-by-hop delay of the SFC f_i does not exceed the end-to-end delay of that SFC;
- $\forall s \in \mathcal{S}, f_i \in F_s, \nu 1 \in f_i, \nu 2 \in \eta(\nu 1) : W_B(\nu_1, \nu_2) \leq W_i^f$. This condition ensures that the hop-by-hop bandwidth of the SFC f_i does not exceed the end-to-end bandwidth of that SFC.

The RAM, the CPU and the DISK of a VNF $g \in \mathcal{G}$ are defined as the sum of all the RAM, the CPU and the DISK of its NFs γ_g, respectively. Formally, $\forall g \in \mathcal{G}$, $\delta_g.RAM = \sum\limits_{\nu \in \gamma_g} \nu.RAM$, $\delta_g.CPU = \sum\limits_{\nu \in \gamma_g} \nu.CPU$, and $\delta_g.DISK = \sum\limits_{\nu \in \gamma_g} \nu.DISK$. Let $W_L(\delta_{g1}, \delta_{g2})$ and $W_B(\delta_{g1}, \delta_{g2})$ denote the latency and the bandwidth between a VNF $g1 \in \mathcal{G}$ and its neighbor $g2 \in \eta(g1)$. $W_L(\delta_{g1}, \delta_{g2})$ is defined as the maximum delay between any two NF neighbors $\nu_1 \in \gamma_{g1}$ and $\nu_2 \in \gamma_{g2}$. Formally, $W_L(\delta_{g1}, \delta_{g2})$ is defined as follow: $W_L(\delta_{g1}, \delta_{g2}) = \underset{\nu_1 \in \gamma_{g1}, \nu_2 \in \eta(\nu_1) \cap \gamma_{g2}}{max} W_L(\nu_1, \nu_2)$. While $W_B(\delta_{g1}, \delta_{g2})$ is defined as the sum of all the bandwidths between all the NF neighbors $\nu_1 \in \gamma_{g1}$ and $\nu_2 \in \gamma_{g2}$. Formally $W_B(\delta_{g1}, \delta_{g2})$ is defined as follow: $W_B(\delta_{g1}, \delta_{g2}) = \sum\limits_{\nu_1 \in \gamma_{g1}, \nu_2 \in \eta(\nu_1) \cap \gamma_{g2}} W_B(\nu_1, \nu_2)$.

Let $\Upsilon(\delta, G)$ a function that randomly selects a set of physical nodes from V, whereby the VNFs δ should be deployed. Let we denote by δ^v the set of VNFs that should be deployed on top of the physical server $v \in \mathcal{V}$. The function $\Upsilon(\delta, G)$ picks up a random subset $\mathcal{V}$ from V ($\mathcal{V} \subseteq V$) that satisfies the following conditions:

- $\forall v1, v2 \in \mathcal{V}, \delta^{v1} \cap \delta^{v2} = \emptyset$. This condition is used to ensure that two physical servers should not host the same VNF;
- $\bigcup\limits_{v \in \mathcal{V}} \delta^v = \delta$. This condition ensures that all VNFs should be deployed in the physical servers;
- $\forall g1, g2 \in \mathcal{G}, (g2 \in \eta(g1)) \implies \exists v1, v2 \in \mathcal{V}, (g1 \in \delta^{v1}) \wedge (g2 \in \delta^{v2}) \wedge ((v1, v2) \in E \vee (v1 = v2))$. This condition is used to ensure that two neighbors VNFs $g1, g2 \in \mathcal{G}$ should be hosted at: $i)$ the same physical server, or $ii)$ two neighbors physical servers;
- $\forall v \in \mathcal{V} : \sum\limits_{g \in \delta^v} \delta_g.RAM \leq v.RAM$. This condition ensures that the RAM used by different VNFs δ^v hosted at the same physical server node $v \in \mathcal{V}$ should not exceed the RAM capacity of v;
- $\forall v \in \mathcal{V} : \sum\limits_{g \in \delta^v} \delta_g.CPU \leq v.CPU$. This condition ensures that the CPU used by different VNFs δ^v hosted at the same physical server node $v \in \mathcal{V}$ should not exceed the CPU capacity of that server v;
- $\forall v \in \mathcal{V} : \sum\limits_{g \in \delta^v} \delta_g.DISK \leq v.DISK$. This condition ensures that the DISK used by different VNFs δ^v hosted at the same physical server node $v \in \mathcal{V}$ should respect the capacity limit of the server v;
- $\forall (v1, v2) \in E : \min\limits_{g1 \in \delta^{v1}, g2 \in \eta(g1) \cap \delta^{v1}} W_L(g1, g2) \geq W_L(v1, v2)$. This condition ensures that the shortest delay between two neighbors VNFs $g1$ and $g2$ (which represents the fastest delay) is bigger than the available delay between their hosted servers $v1$ and $v2$, respectively;
- $\forall (v1, v2) \in E : \sum\limits_{g1 \in \delta^{v1}, g2 \in \eta(g1) \cap \delta^{v1}} W_B(g1, g2) \leq W_B(v1, v2)$. This condition ensures that the sum of the bandwidth between all the VNFs neighbors $g1$ and $g2$ hosted at the servers $v1$ and $v2$, respectively, should not

exceed the bandwidth between the servers $v1$ and $v2$.

The function $\Upsilon(\delta, G)$ returns a true if it could succeed to select the set of physical nodes that satisfy the above conditions. Otherwise, it will return false. Initially, Algorithm 2 generates the set of NFs γ from different network slices and their SFCs (Algorithm 2: Lines $1-8$). Then, the number of VNFs is initiated to equal the number of NFs γ (Algorithm 2: Line 9). In the worst case scenario, each NF $\nu \in \gamma$ should run on top of a VNF $g \in \mathcal{G}$. The number of repetitions $\mathcal{R}$ is initiated with $\mathcal{K}$ (Algorithm 2: Line 10). Then, GDMK heuristic enters in an infinite loop to create different VNFs δ and place them at different physical servers. In the loop, initially, the Algorithm randomly generates the set of VNFs δ that satisfies the aforementioned conditions, such as the number of VNFs δ less or equals to $\mathcal{N}$ (Algorithm 2: Line 12). Note that, initially, the number of NFs γ equals the number of VNFs $\mathcal{N}$ (i.e., $|\delta| = |\gamma|$). Then, during the processing phase of the proposed Algorithm, the number of VNFs $\mathcal{N}$ will be reduced. After that, the Algorithm places different VNFs δ in G, and checks if the aforementioned conditions hold (Algorithm 2: Line 13). If the conditions hold (i.e., The function Υ returns true) and the size of the new VNFs $\delta_{\mathcal{T}}$ is better than the best-achieved solution, then the new best configuration of VNFs would be moved from temporal configuration $\delta_{\mathcal{T}}$ to the best configuration δ (Algorithm 2: Lines $14-16$). Also, the number of VNFs and repetition are re-initialized. In the case that the Algorithm does not succeed to place the VNFs in G, the number of repetitions is reduced by one (Algorithm 2: Line 19). Thus, the Algorithm checks if the number of repetitions reaches zero (Algorithm 2: Line 20). If so, the Algorithm also checks if the best configuration exists. In case of success, the Algorithm terminates, otherwise, the Algorithm keeps seeking for the best VNFs configuration.

A. Complexity Analysis

Analyzing the complexity allows us to forecast the computational time, nb_{slice}, nb_{sfc}, nb_{nf} are chosen to represent the number of network slices, the number of SFCs and the number of NFs, respectively. During the initialization phase (Algorithm 2: Lines $1-8$) we have a complexity equal to the multiplication of those three defined variables (nb_{slice} * nb_{sfc} * nb_{nf}). Next, in the main loop part of the proposed algorithm, the proposed solution will have to execute ($\mathcal{K}$ + 1) * (nb_{nf} + nb_{dcs}) in the best case scenario. While the worst case complexity will be X * ($\mathcal{K}$ + 1) * (nb_{nf} + nb_{dcs}), with "X" to represent the number of defective solutions found before the effective one that will be executed $\mathcal{K}$ additional times to be committed as the best solution.

VI. Performance evaluation

In the simulation, we have used Python and the Gurobi optimizer to implement our optimization framework to solve the previously formulated problem. Meanwhile, the proposed heuristic is implemented using the Python language. The underlying layer's components (i.e nodes, edges, CPU, RAM, Disk), network slices, SFCs, NFs, and the NFs' resources requirements are randomly generated to simulate a more realistic environment. The resource demands of each NF in terms of both, bandwidth and latency between NFs, follows a discrete uniform distribution over the interval $[50, 100]$.

We conducted our experiments on a multi-core server as described in Table I.

TABLE I: Hardware Configuration.

Type	Configuration
CPU	Dual Intel Xeon E5-2680 v3 @ $2.5GHz$
Memory	$256GB$
Linux	Ubuntu 16.04
Kernel	4.4.0-72

We started the evaluation of our solution's behavior by varying the number of network slices and NFs. For each experiment, we operate 100 repetitions, changing the underlying layer's components deployment and compute the number of nodes' used as well as the computational times. Afterward, we present the mean and 95% Confidence Interval of the number of nodes' used in the proposed architecture and the computational cost in seconds. It is noticed that for the Fig. 3, Fig. 4, and Fig. 5, the number of nodes used and the computational time of the evaluation are represented by the pentagon shape and the hexagon shape, respectively for the optimal solution. While the circle shape shows the number of used nodes and the star shape the computational time for the heuristic solution.

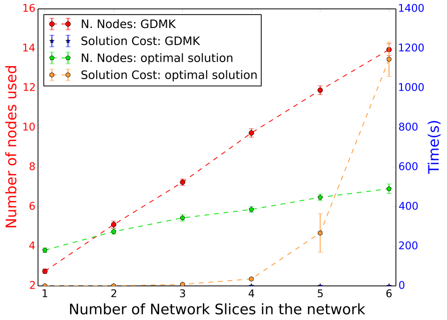


Fig. 3: Varying the number of network slices from 1 to 6.

Fig. 3 features the comparison between the optimal solution (OCNSD) and the proposed heuristic (GDMK), we vary the number of network slices over both the VNF layer and the physical network from 1 to 6, while keeping for each slice both the number of SFCs and NFs constant, i.e., running our simulations with 1 and 4, respectively. In Fig. 3, the left Y-axis represents the number of nodes used, the right Y-axis shows the required time in seconds to solve the optimization problem, and the X-axis portrays the number of network slices in the proposed architecture. For the optimal solution, the number of used nodes represented by the pentagon shape increases linearly from 4 to 7 when the number of network slices increases. It is noticed that the number of used nodes follows a binary logic which means real values must be rounded up. The mean number of nodes activated is 5.5387 with a standard deviation of 1.0417. Meanwhile, the computational cost showed as a hexagonal shape increases exponentially when varying the number of network slices. The heuristic solution (GDMK) shows that the number of active nodes, represented by the circle shape, increases from 3 to 13 when the 6th variation of network slices is reached. The mean number of nodes activated is 8.44 with a standard deviation of 3.8493. Concurrently, the computational cost, portrayed by the star shape, increases linearly from $0.0036s$ to $0.01992s$. The linear regression parameters are 0.0033 and 0.0003, as α and β, respectively, for the solution cost of our set of experiments.

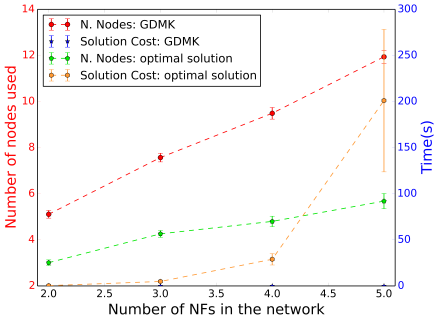


Fig. 4: Varying the number of NFs from 2 to 5.

In Fig. 4 a comparison between the optimal solution (OCNSD) and the proposed heuristic (GDMK) is illustrated when varying NFs number in the network from 2 to 5 while fixing the number of network slices and SFCs in the network to two instances for each. We kept the same representations as before for the Y-axes except for the X-axis in which we are considering the number of NFs in our network instead of network slices. Regarding the optimal solution, we observed a linear increase concerning the number of activated nodes, represented by the pentagon shape, in the network, with a mean number of activated nodes of 4.4375 and a standard deviation of 0.9675. Similar to the previous experiment we can perceive an exponential growth in the computational time (hexagonal shape). While for the heuristic approach, we observed that the number of activated nodes represented by the circle shape, increases from 5 to 12 with a mean number of activated nodes of 8.5275 and a standard deviation of 2.5084. Still, in Fig. 4, we noticed that the solution cost (the star shape) in seconds increases linearly with the number of activated nodes in the network: the linear regression of these samples was 0.0035 and -0.0012, as α and β, respectively.

The experiment in Fig. 5 shows how the underlying topologies impact both the optimal solution (OCNSD) and the proposed heuristic (GDMK). We vary the number of physical hosts belonging to the physical layer in our proposed architecture while fixing the number of network slices, SFCs, NFs. Concerning the optimal solution, the first observation that we can draw from this figure is that the number of used nodes, represented by the pentagon shape, stagnates in 5, even when increasing the number of physical hosts 5 fold. Moreover, we clearly observe the exponential behavior of the computational time, expressed by the hexagon shape, which allow us to conclude that when we increase the number of underlying nodes, the number of links between those nodes increases due to the higher density of nodes' in our network which will raise the probability of deployment of the network slices causing by the same time the exponential growth in the computational time (optimal solution search always best options). For the heuristic solution, the same stagnation in terms of the number of used nodes, represented by the circle shape, is observed with a higher number of the used nodes (6 for the heuristic instead of 5 for the optimal solution). However, the computational time for the heuristic solution is linear and has the following linear regression samples 0.0096 and 0.00099, as α and β, respectively.

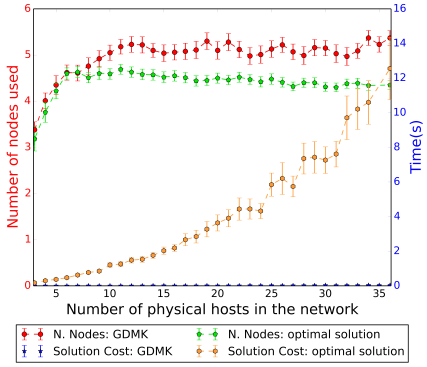


Fig. 5: Varying the number of physical hosts from 1 to 36.

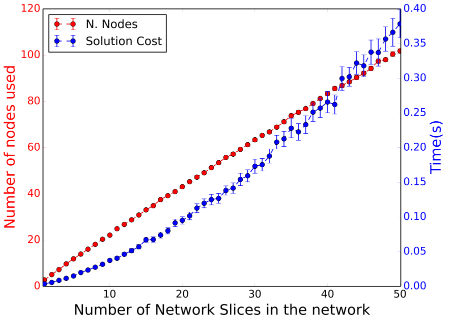


Fig. 6: Varying the number of network slices from 1 to 50 in large scale networks.

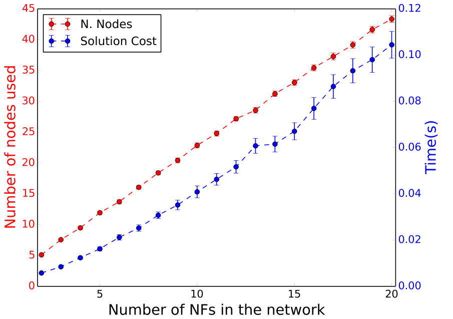


Fig. 7: Varying the number of NFs from 2 to 20 for large scale networks.

Finally, we evaluated the computational cost of the heurestic GDMK by varying the number of slices and SFCs. Fig. 6

depicts the impact of the number of network slices on the proposed heuristic GDMK in terms of the cost (i.e., number of used nodes) and the execution time. In this experiment, we have varied the number of sliced from 1 till 50, while keeping for each slice both the number of SFCs and NFs constant. The results plotted in Fig. 6 shows that both the cost and the execution time increases linearly with the number of slices in the network. While Fig. 7 depicts the performances of the proposed heuristic by varying the number of NFs in the network from 2 to 20 while fixing the number of network slices and SFCs in the network by 2. The results plotted in Fig. 7 show clearly that the cost and execution time increase linearly with the number of NFs.

VII. Conclusion

In this article, we presented a detailed modeling of the cross-domain network slicing, we introduced a novel cost optimal deployment of network slices and evaluated our proposal using multiple network topologies, and in particular, focused on following the standard specifications pronounced by the 3GPP. Interesting results were obtained where varying the number of network slices and NFs, we observed that the computational time grows exceptionally. A heuristic algorithm was designed, implemented and evaluated to compute the underlying node distribution in polynomial time. In future works, the mobility of Network Slices will be investigated to fulfill mobile users' requirements and allow a better QoE/QoS, the exploration of real-world NFV orchestrator like OSM, ONAP, Tacker (OpenStack's NFV components) will be considered in order to integrate both the designed algorithms (optimal and heuristic) and provide information to the NFV orchestrator to determine the best place to instantiate NFs.

Acknowledgment

This work was supported in part by the Academy of Finland Project 6Genesis Flagship (grant no. 318927), and in part by the European Unions Horizon 2020 Research and Innovation Program through the MATILDA Project under Grant No. 761898.

References

[1] R. A. Addad, T. Taleb, M. Bagaa, D. L. C. Dutra, and H. Flinck, "Towards Modeling Cross-Domain Network Slices for 5G," in *2018 IEEE Global Communications Conference, IEEE GLOBECOM*, Abu Dhabi, UAE, Dec 2018.

[2] N. Alliance, "5G white paper," Tech. Rep., February 2015. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

[3] G. P. A. W. Group, "View on 5g architecture," Tech. Rep., July 2016. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-For-public-consultation.pdf

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing; softwarization: A survey on principles, enabling technologies; solutions," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2018.

[5] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[6] 3rd Generation Partnership Project (3GPP), "Release 15," 2018, (Last visit on : 02-05-2018). [Online]. Available: http://www.3gpp.org/release-15

[7] ——, "Release 16," 2018, (Last visit on : 02-05-2018). [Online]. Available: http://www.3gpp.org/release-16

[8] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, no. C, pp. 138–155, Nov. 2016. [Online]. Available: https://doi.org/10.1016/j.jnca.2016.09.001

[9] 3rd Generation Partnership Project (3GPP), "3gpp 5g core network status," Tech. Rep., November 2017. [Online]. Available: http://www.3gpp.org/ftp/information/presentations/Presentations_2017/webinar-ct-status-11-2017.pdf

[10] E. Joe Wilke, "5g network architecture and fmc," Tech. Rep., July 2017. [Online]. Available: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/201707/Documents/Joe-Wilke-%205G%20Network%20Architecture%20and%20FMC.pdf

[11] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 418–423.

[12] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, April 2015.

[13] H. Ko, D. Suh, H. Baek, S. Pack, and J. Kwak, "Optimal placement of service function in service function chaining," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2016, pp. 102–105.

[14] X. Song, X. Zhang, S. Yu, S. Jiao, and Z. Xu, "Resource-efficient virtual network function placement in operator networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[15] S. Jiao, X. Zhang, S. Yu, X. Song, and Z. Xu, "Joint virtual network function selection and traffic steering in telecom networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[16] H. Li, L. Wang, X. Wen, Z. Lu, and L. Ma, "Constructing service function chain test database: An optimal modeling approach for coordinated resource allocation," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[17] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with nestor," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 91–105, March 2017.

[18] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[19] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 50–56.

[20] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486012

[21] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486019

[22] 3GPP, "System Architecture for the 5G System; Stage 2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 03 2018, version 15.1.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[23] A NOvel Radio Multiservice adaptive network Architecture for the 5G era, "NORMA," 2018, (Last visit on : 02-11-2018). [Online]. Available: http://www.it.uc3m.es/wnl/5gnorma/index.html

[24] L. T. Kou and G. Markowsky, "Multidimensional bin packing algorithms," *IBM J. Res. Dev.*, vol. 21, no. 5, pp. 443–448, Sep. 1977. [Online]. Available: http://dx.doi.org/10.1147/rd.215.0443