ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture / Shinde, Swapnil Sadashiv; Marabissi, Dania; Tarchi, Daniele. - In: COMPUTER NETWORKS. - ISSN 1389-1286. - ELETTRONICO. - 201:24 December 2021(2021), pp. 108598.1-108598.15. [10.1016/j.comnet.2021.108598]

Availability:
This version is available at: https://hdl.handle.net/11585/837711 since: 2022-03-11

Published:

DOI: http://doi.org/10.1016/j.comnet.2021.108598

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

28 April 2024

A Network Operator-biased approach for Multi-Service Network Function Placement in a 5G Network Slicing Architecture

Swapnil Sadashiv Shinde[a], Dania Marabissi[b], Daniele Tarchi[a,*]

[a]*Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Viale del Risorgimento, 2, 40136, Bologna, Italy*
[b]*University of Florence, Via Santa Marta, 3, Firenze, 50139, Italy*

Abstract

The 5G communication standard is characterized by an increased softwarization, allowing a higher flexibility able to cope with different requirements and services. In particular, Network Function Virtualization (NFV) is a recently introduced technology that enables a software implementation of different network functions exploiting virtualization techniques, hence, enabling their flexible deployment upon system requirements. Boosted by NFV, the concept of network slicing is gaining great attention in 5G networks. The idea is that physical communication and computing resources are sliced in multiple end-to-end logical networks, each one tailored to best support a specific service. The advantages of NFV, in the network slicing context, are even more evident in distributed computing environments, such as the edge-to-cloud continuum, recently introduced for enabling a flexible deployment of multiple functions. In particular, thanks to the introduction of cloud-native technologies, based on the usage of containerization and microservice technologies, the virtual network functions (VNFs) deployment and their orchestration is an easy operation, allowing the on-the-fly network configuration. Gaining from the NFV, Network Slicing and Edge-to-Cloud continuum paradigms, we propose a new network function allocation problem for multi-service 5G networks, able to deploy network functions on a distributed computing environment depending on the service requests. The proposed approach jointly considers Radio Access Network (RAN) and Core Network (CN) functions and, differently from other approaches, introduces an option able to bias the function placement depending on the service requirements, allowing a fast-and-easy operator-side deployment of the network functions. We propose to solve the problem through a Genetic Algorithm able to approach the optimal solution but with reduced complexity and execution time. The performance is compared with two other heuristic algorithms and with an exhaustive search algorithm, introduced as benchmarks, showing the benefits of the selected solution in terms of performance, flexibility and complexity.

Keywords: 5G, Network Slicing, Edge Computing, Network Function Placement, Genetic Algorithms

1. Introduction

5G technology aims to create a fully digital connected society by enabling a large number of network services and applications. The 2021 Ericsson Mobility Report forecasts that by 2026 there will be 3.5 billion of 5G subscriptions, accounting for 40 percent of all mobile subscriptions [1]. Consequently, an increasing number of heterogeneous devices such as smartphones, tablets, wearable, sensors, will require pervasive connections and a wide range of services characterized by different requirements in terms of performance (e.g., latency, data rate, reliability), and network functionalities (e.g., security, mobility, radio resource management).

In order to support this high variety of requirements, a flexible network architecture with reduced time-to-market for new services is required. Toward this goal, network softwarization is an emerging key approach in 5G systems, that uses technologies like Network Function Virtualization (NFV), and Network Slicing for providing programmability, flexibility, and modularity in different parts of the network [2]. NFV decouples network functions from proprietary hardware (HW) and runs them as software instances through proper virtualization techniques [3], thus overcoming the lack of flexibility of traditional HW-based network functions. Network slicing allows to create multiple end-to-end logical networks, through the composition of chains of physical and virtual network functions (VNFs), to run on top of a common physical network infrastructure [2]. Each logical network is tailored to provide specific services and/or a particular tenant with a certain level of guaranteed network resources.

The 5G network architecture is composed of several functions that can be roughly grouped into Radio Access Network (RAN) and Core Network (CN) functions. While CN functions are historically placed in centralized nodes, it is only in the recent years that an effort has been done for moving all or part of the RAN functions to a central point in order to enhance the network management. Indeed, at RAN level, flexibility and softwarization are supported by the Centralized or Cloud-Radio Access Network (C-RAN) concept that allows to exploit cloud technologies in the access network [4]. Moreover, Multiaccess Edge Computing (MEC) has been recently identified as a 5G-RAN key technology for supporting services with reduced la-

*Corresponding author
Email addresses: swapnil.shinde2@unibo.it (Swapnil Sadashiv Shinde), dania.marabissi@unifi.it (Dania Marabissi), daniele.tarchi@unibo.it (Daniele Tarchi)

tency and massive access [5]. Through MEC, mobile operators can deploy a set of nodes equipped with computational and storage capacity in the RAN, bringing cloud services closer to users, thus reducing delays in the process and backhaul overload. However, MEC computational/storage capacity is limited if compared to a central cloud, thus in 5G systems the two approaches can be paired to accommodate emerging services. MEC can be integrated with the classical Cloud Computing approach leading to a new distributed scenario, often referred to as Edge-to-Cloud continuum [6], where the processing capabilities are distributed along the path from the user to the central cloud [7].

Starting from NFV, Network Slicing and Edge-to-Cloud continuum paradigms, this paper proposes a new solution for a flexible placement of network functions in network elements, for supporting different service classes. The fast-and-easy proposed solution can cope with the complexity of the 5G scenario, aiming at minimizing a properly defined preference-weighted cost, while respecting network and services constraints. In particular, network nodes are organized in multiple hierarchical layers and the optimization of the network functions placement is performed in a 3D scenario where services (i.e., slices), functions and layers are jointly considered. We propose to use a Genetic Algorithm (GA) that almost approaches the optimal solution in a reasonable amount of time. The proposed solution takes into account a bias factor in the function placement, later named as *preference*, enabling the possibility of favouring some function deployments specific for each slice, while keeping the problem simple. This factor allows a function deployment based on the operators view and experience.

1.1. Related Literature

Network slicing and VNFs placement problems are widely addressed in the literature, but very often the focus is only on one of the two concepts without considering the close relationship with the other. For example, several existing works focus on basic slicing concepts and design, key enablers and challenges as [2, 8, 9] without proposing VNFs placement solutions. Similarly, the resource allocation problem (i.e., capacity, computation and storage) among slices has received a wide attention, as in [10–14]; however, in this context, some papers consider only spectrum resources (e.g., [10, 11]), while others take into consideration also the computing resources requested by VNFs whose placement is, however, fixed or not considered (e.g., [12, 13]). In [14], the authors survey and analyse several resource allocation solutions. At the same time several papers address the problem of providing an efficient VNFs placement with different goals, such as reducing the computation and communication resource wastage, increasing the energy-efficiency or the network throughput while reducing the end-to-end delay, although without considering the network slicing concept and the presence of heterogeneous services [15–17]. In [18] a survey on resource allocation and VNFs placement problems is provided.

Since network slices can be deployed as a chain of VNFs, the two concepts are strictly related and have to be jointly investigated. Consequently, VNFs placement problem over heterogeneous network slices with different objectives and constraints is gaining attention [19–22]. In [19], the authors study the optimal deployment of VNFs and allocation of computational resources in a hybrid two-clouds C-RAN infrastructure, considering different 5G service requirements and specific 5G RAN functions. The goal is to minimize the overall used computational resources by imposing constraints on VNFs. The problem is reformulated as an integer linear problem (ILP) and solved through a standard solver. Similarly, [20] deals with the optimal VNFs placement problem in C-RAN. VNFs are placed over virtual resources spread across multiple clouds, composing a centralized base-band unit (BBU), with the goal of minimizing latency and cost. The proposed solutions are based on the branch-and-bound and simulated annealing methods. However, in this paper the network slicing concept is not analysed in depth, and service constraints are not considered: only different data rates requested by different services are taken into account. The NFV problem in a C-RAN infrastructure is also considered in [21] where a formulation of a C-RAN system that deploys VNFs on an edge data center is presented analysing six different requirements. Here, different network slices with heterogeneous requirements/constraints are considered, while VNFs are placed in the edge data center. VNFs placement, traffic routing and resource allocation in multiple network slices while considering VNFs sharing is investigated in [22]. The aim is to minimize the amount of required bandwidth and computational resources; a heuristic solution is proposed for solving the problem.

Different problems are addressed in [23, 24]. In particular, [23] studies VNFs placement when VNFs are decomposed into multiple-sub functions that can be reused by different network slices to reduce the substrate network cost. Network slices and functions are non specialized, and slices constraints are not considered. Conversely, in [24] the goal is to minimize the reliability degradation and the cost, while respecting the delay constraint when parallel VNF processing is considered. The solution is based on a Tabu-search algorithm. Here, only one delay-constrained slice is considered.

1.2. Paper Contribution

As previously shown a vast literature exists on network slicing and VNFs placement, however:

- in many papers the two concepts are separately investigated even if they are strictly related, as for example in [2, 8–13, 15–17];
- the majority of papers jointly considering slicing and VNFs placement focuses on a hybrid RAN infrastructure with a central cloud and several edge clouds that provide computational capacity to execute functions closer to the users [19–21, 24]. Only a few papers consider both CN and RAN functions [22, 23];
- existing papers often propose high-level system models, where a set of generic functions can be executed on a set of generic nodes [21–24], while only a few consider a real 5G network, with different network elements and specific

5G functions with their characteristics and constraints, and mainly focused on RAN [19, 20];

- even when network slices are considered in the system modelling, many papers do not impose specific constraints and requirements on services and VNFs, while consider generic heterogeneous services, characterized by a certain data-rate [20], or consider only one slice, while no coexistence of multiple slices [24].

Differently from previous approaches, in this paper we propose a model that considers the VNFs placement problem supporting different network slices characterized by specific requirements and constraints. The paper takes into account the coexistence of multiple slices on the same physical network, and, hence, the cross-slice impact. The considered architecture is end-to-end (E2E), including both RAN and CN functions as well network elements, specifically referring to the 5G architecture. Moreover, the E2E infrastructure includes MEC facilities that are generally considered only when RAN is analysed, so that the Edge-to-Cloud continuum is considered.

Finally, the proposed approach introduces the concept of *preference* that is not present in the literature, at the best of our knowledge. The idea is that of a bias factor that can be managed by the operator (e.g., depending on its own view and experience, and/or on business revenues) allowing to pick some function deployments specific for each slice, while keeping the problem simple. Moreover, the preference can also be used to impose some constraints on VNFs deployment (e.g., the Radio Frequency functions cannot be deployed in the central cloud).

The main contributions of this paper can be summarized in some key aspects:

- The VNF placement problem in the multi slice-multi layer E2E architecture has been properly modeled through a *preference weighted cost function*. Constraints given by service requirements and limitations of the physical infrastructure are considered.
- The cross-slice impact is considered and analysed by showing the effect on the performance of a slice when varying the parameters of another slice.
- The proposed problem jointly considers RAN and CN functions, and is tailored for an environment that considers the new Edge-to-Cloud continuum paradigm. It is analysed and shown how the E2E functions placement varies in different network configurations.
- We propose a GA as a potential solution method for the defined problem, allowing to approach the optimal solution to the placement problem while keeping the complexity reduced.
- The proposed GA solution is compared with two simple heuristic approaches, where VNFs placement depends upon the preference values associated with them. We have also considered the Exhaustive Search Algorithm (ESA) to achieve the optimal solution and verify the accuracy of the proposed GA method.
- Performance evaluation has been carried out in several multi-service scenarios where different parameters' settings are considered for understanding the potentialities of the approach in a realistic environment. In particular, slices' key performance is shown as well as the computational load and the function placement in different network elements.

2. Proposed Network Model

We focus on a multi-layer distributed C-RAN architecture where several remote radio heads (RRHs) units are connected with a BBU pool, that, in turn, is connected to the Edge Computing layer, composing a service area, as depicted in Figure 1. A high speed backhaul connects the service areas to a Cloud Infrastructure. In this paper we consider a two-layers Cloud Infrastructure, where a Network Operator Cloud (NOC) is used for operator specific operations, while a multi-tenant Cloud facility is supposed for multi-operator applications deployment. The high-level network architecture is depicted in Figure 1, where it is possible to see different RRHs positioned as small-cell access points, several BBUs, connected through a high speed Common Public Radio Interface (CPRI) interface, as well multiple Edge facilities acting as processing nodes. A two-layers Cloud facility is also depicted considering both the NOC and the multi-tenant Cloud infrastructure. Table 1 reports the list of symbols used in what follows.

Table 1: Symbols list.

Symbol	Description
L	number of hierarchical network layers
F	number of VNFs
S	number of slices
$\mathcal{F}$	set of VNFs
$\mathcal{F}_{\mathcal{CP}}, \mathcal{F}_{\mathcal{UP}}$	set of CP and UP VNFs
$\mathcal{S}$	set of slices
$p(f,l,s)$	preference of placing f-th function of the s-th slice on l-th layer
r_s^u, r_s^c	UP and CP data generation rates per user of the s-th slice
$\eta'(f,s)$	cost of placing the f-th function of the slice s-th
$\eta(f,s)$	preference weighted cost of placing the f-th function of the slice s-th
$U_t(s)$	average number of active users of the slice s-th
U_{t_s}	total number of users of the s-th slice
$pdf_s(\cdot)$	distribution of the users activity of the slice s-th
$\mathbf{A}$	function placement matrix whose element is $a(f,l,s)$
O_l	maximum data processing capacity of the l-th layer
γ	number of FLOPs required to process a single bit
τ_{UP_s}, τ_{CP_s}	maximum allowed delay of s-th slice in UP and CP
$\delta_{UP_s}, \delta_{CP_s}$	latency between the user and the UP/CP end-point for the s-th slice

The previously described network elements can be supposed as logically organized in L hierarchical layers, where nodes have increasing processing and communication capabilities.

The network functionalities can be implemented through the deployment of a set $\mathcal{F}$ of F VNFs, to support a set $\mathcal{S}$ of S network slices each one characterised by a different type of service with specific requirements. Network nodes and communication

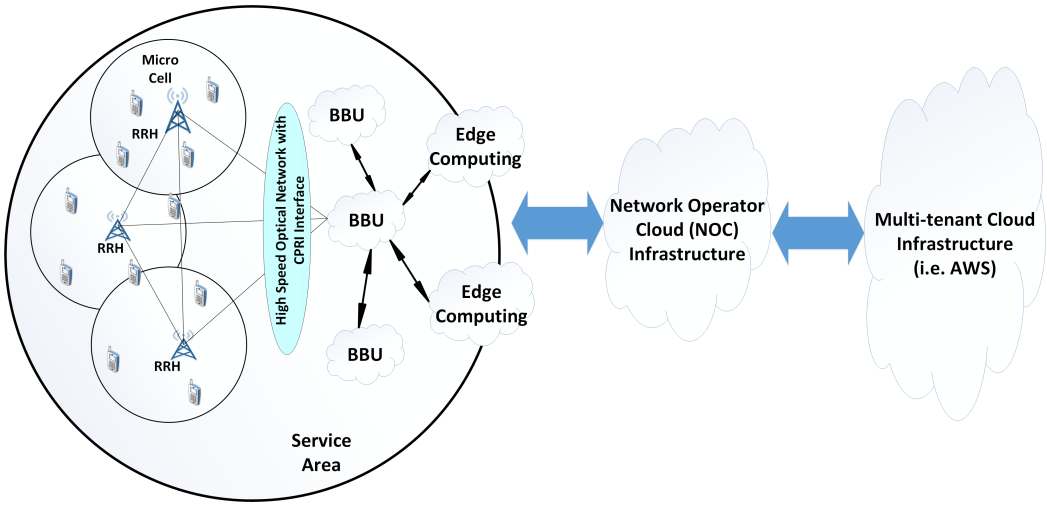

Figure 1: The multi-service distributed 5G network architecture.

links are resource-constrained. Each function $f \in \mathcal{F}$ is characterized by the amount of computing load required for its execution, as detailed later. In this paper we resort to the availability of cloud-native solutions. While the NFV concept resides on the general idea of software functions, its implementation can have different alternatives. To this aim we assume that each function is available through containers and/or microservices implementations, so as to be easily deployed and moved from different layers, hence, allowing an on-the-fly network configuration [25].

Even if NFV approach ideally allows an arbitrary function deployment, some constraints should be considered. As an example, it is common understanding that some functions, e.g., base band processing should be located nearer to the users while others, e.g., packet filtering, could also be centralized. At the same time defining a proper rule for placing the function would be very complex, since users and operators could have conflicting goals. To this aim we resort to the definition of a *preference function*, $p(f, l, s)$, that allows to bias the placement of function f on the layer l when implementing the slice s. In particular, $p(f, l, s) \in [0, 1]$ and

$$\sum_{l=1}^{L} p(f, l, s) = 1,$$

where a higher $p(f, l, s)$ corresponds to an advantage when placing the function f of the s-th slice on layer l. If $p(f, l, s) = 0$, function f cannot be placed on layer l, while $p(f, l, s) = 1$ means that function f has to be placed on layer l; any other intermediate value can be used for mapping a different behaviour in placing the f-th function. Consequently, through the definition of a simple and effective weight, i.e., the preference, it is possible to flexibly define where a given function should be placed in a distributed architecture for a given slice. In particular, the operator can exploit its know-how when a certain service should be deployed. Indeed, a proper fine-tuning of the preferences allows to set the behaviour of the function placement so that each operator is able to drive the placement in the preferred way while respecting the system constraints. Moreover, such an approach allows to set a specific weight for placing the VNFs, also considering business and operator-driven choices.

2.1. Multi-slice Multi-layer Architecture

In this section we define the system elements considered in this paper. We focus on a multi-layer architecture, supporting main 5G network functions, and able to implement three different classes of service classes (enhanced mobile broadband -eMBB-, ultra reliable low latency communications -URLLC-, and massive machine type communications -mMTC) through proper network slices. It is worth to be noticed that the following parameters allow to define a reference scenario, that can be changed without loosing the validity of the presented model and of the solution that will be presented later.

2.1.1. Layers

The considered network architecture is organized in 5 layers: the first two layers represent the RAN working at the network edge, then a Central Operation layer follows (i.e, the Edge Cloud), while the farthest layers represent the Network Operator Data Center (i.e., the NOC) and a multi-tenant Cloud infrastructure (e.g., Amazon Web Service (AWS)). The layers are hierarchically organized, and several nodes of layer l can be aggregated under the same node of the layer l+1. RRHs are placed at layer 1 (L1) and are connected with a BBU pool at layer 2 (L2), able to handle several cells corresponding to the antenna elements associated with. In order to take into account the variety of network deployment and services that must be supported by 5G, the layer L2 is further split in two sub-layers (L2a and L2b); indeed, in some contexts (e.g., a macrocell where RRH and BBU are co-located, or in case of functions that do not need coordination among different antenna sites) and/or for some services (e.g., URLLC services requiring very small delays), it is possible to move part of the BBU processing near to the RRH (L2a). One or several BBU-sites are then connected to an

Edge Cloud (L3) where some functions can be executed and, in turn, one or more Edge Clouds are connected to the NOC (L4), where the Network Operator functions are executed. Following this a multi-tenant Cloud infrastructure (L5) is considered, possibly deployed through a commercial cloud solution. Each layer is connected to the next through proper high data-rate links. In particular, while end-devices are connected to the RRH units through proper wireless technologies, RRH and BBU-site are supposed to be connected through a high data-rate fiber link over CPRI. The backhaul links between Edge cloud, centralized NOC and multi-tenant Cloud infrastructures are instead supposed to use high-speed dark fiber links [26].

2.2. Functions

In order to have a proper modeling of the slices, we resorted to the usage of the 5G RAN and CN functions. For the purpose of this work, we consider the most commonly used functions.

5G RAN Functions Referring to the 5G RAN functions introduced in the 3GPP standard [19], we consider that 8 possible RAN functions can be deployed on the network nodes, i.e., radio function (RF), lower physical layer (PHY), higher PHY, lower medium access control (MAC), higher MAC, lower radio link control (RLC), higher RLC, packet data convergence protocol (PDCP). These functions form a VNF chain and can be placed at different layers exploiting virtualization technologies. However, implementing these functions individually over several platforms can be costly in terms of communication overheads. On the other hand, placing all of them together on the same platform can reduce the flexibility. Hence, following the approach proposed in the 5G standard [27], we consider that RAN functions can be organized in three main logical subsets, i.e., radio unit (RU), centralized unit (CU) and distributed unit (DU). Multiple CU-DU split options are introduced, providing extra flexibility into RAN [27]. In particular, possible split options have been numbered and suggested to be used for specific services, as for example in [28], where Option 5 is preferred for the eMBB slice, while Option 2 is considered for the case of URLLC and mMTC slices. Since split options can be selected based on the service type and requirements, without loss of generality, in this work, we consider Options 2, 6, and 8. In the split Option 2 PHY, MAC, and RLC functions are in the DU, while PDCP is located on CU; in the split Option 6, only PHY functions are in DU while the remaining functions (MAC, RLC, and PDCP) are on CU. Finally, in case of split Option 8, all RAN functions are on CU, forming a highly coordinated RAN architecture [29]. It should be noted that the selected options are only for implementation purposes, and our work can be extended for any possible functional split option. In the following the RAN functions associated with the selected split options are described, where for simplicity we do not consider lower and upper split of each function:

- **RF** includes the analog-to-digital and digital-to-analog functionalities performed on RRH during uplink and downlink communications.
- **PHY** performs the digital signal processing during uplink and downlink communications between user equipment (UE) and gNB.
- **MAC** provides low-level control functions for the PHY layer through data transmission scheduling between UE and gNB.
- **RLC** handles the functions related to reliable data delivery, segmentation, etc.,
- **PDCP** performs higher-level transport functions, including header compression and security.

5G CN Functions For describing the functions performed at 5G-CN we refer to the legacy 4G/LTE Evolved Packet Core (EPC) network architecture whose main network functions are the Mobility Management Entity (MME), the Home Subscriber Server (HSS), the Serving Gateway (S-GW), the Public Data Network Gateway (P-GW), the Policy and Charging Rule Function (PCRF), the Traffic Detection Function (TDF) and the Authentication, Authorization, and Accounting (AAA) server. When 5G has been introduced the previously listed functions have been rearranged, while others have been added. Here we describe the main UP and CP network functions in the 5G CN that are used in this paper:

- **UPF** UP Function includes UP functionalities of P-GW and S-GW as packet inspection, routing and forwarding, traffic usage reporting, QoS handling, uplink traffic verification, transport level packet marking in the uplink and downlink, downlink packet buffering, data notification triggering, etc.
- **AMF** Access and Mobility Management Function includes some MME functionalities of EPC such as digital side termination of RAN signaling, access authentication, registration, connection, reachability, and mobility management for the CP signaling.
- **AUSF** Authentication Server Function includes the HSS and AAA server functionalities of EPC. It supports authentication for 3GPP access and untrusted non-3GPP access.
- **SMF** Session Management Function includes CP functionalities of S-GW, P-GW, and some MME functions of EPC, including directing traffic flows, acting as an anchor point for setting up, modifying, and tearing down networking sessions across different parts, etc.
- **NSSF** Network Slice Selection Function helps to provide network slicing functionalities including selecting the set of network slice instances serving the UE, providing support for network slice restriction, and network slice instance restriction.
- **NEF** Network Exposure Function acts as a proxy providing a well-defined API allowing operators and other

users to interconnect with the network, and to access information about the type of services available, statistics and analytic information within the network, etc.

NRF Network Repository Function stores the network function (NF) profile of available NF instances and their supported services including the third-party applications, available for exposure through NEF to other operators and users.

PCF Policy Control Function performs the PCRF functionality of EPC, which includes prioritizing different types of traffic flows over the network. It supports a unified policy framework to govern the network behavior and implements policy rules to CP functions.

UDM Unified Data Management performs all key management functionalities over the network including user Identification Handling, access authorization based on subscription data, UE's Serving NF Registration Management, subscription management, etc.

A possible network functions placement in the considered architecture is represented in Figure 2, where it is possible to notice that it may depends on the considered network slice, as later shown in the numerical results.

We want to underline that also application functions could be added to the list and managed using the proposed framework. However, we limit our study to network functions, because the possible applications that can be deployed on a 5G network are almost infinite.

2.2.1. Slices

We consider three slices (i.e., $S = 3$), corresponding to three broad service categories identified by the ITU [30]:

eMBB provides high data rate and low-latency links to make the mobile services to step a level up, by allowing always-on always-connected mobile terminals with real-time reactivity. Seamless coverage and high mobility are needed in wide-areas with higher data rates than today, while high user density, and very high traffic capacity are needed in hotspots.

mMTC is one of the most expected 5G-enabled services, giving the possibility to interconnect a high density of devices (i.e., hundreds of thousands of devices per square kilometer). Literally, everything can radiate and be connected. Typically these devices are characterized by a relatively low volume of data to transmit and stringent power consumption constraints; moreover, they must be low cost.

URLLC necessitates very stringent requirements in terms of latency (i.e., E2E delay around 1 ms), reliability (i.e., packet error probability 10^{-5} or lower) and availability. The focus is on critical applications as autonomous vehicles, tactile internet-based remote control, healthcare, and mission-critical Internet of Things (IoT), where timely and accurate communications impact safety: monitoring and control must occur in real-time.

2.2.2. Preferences

The proposed idea considers that the placement of each function can be mapped through a proper weight, modeling the preference for placing that function on a specific layer for a given service/slice. Preferences could be considered as a feasibility weight introduced by the operator for biasing the function placement when matching the user request with the operator needs. To this aim, different values can be set for each slice depending on specific requirements and characteristics [31], while each network slice is composed of functions according to service needs.

In general, the eMBB slice requires high data rate and high spectral efficiency, hence would benefit from multi-connectivity, inter-cell cooperation (e.g., joint TX/RX, enhanced inter-cell interference coordination, coordinated scheduling); this implies that the higher layer functions of the RAN protocol stack and CN functions would be preferably more centralized. Moreover, eMBB traffic source and destination can be diverse, hence, a centralized approach would be preferable. To this aim, as an example, SMF is preferably placed on the Cloud.

In contrast, URLLC applications are mainly driven by the need of low-latency while data rate is in general limited. The tight latency requirements for this slice push towards its implementation as close to the users as possible, implying that even the PHY, MAC, RLC and PDCP functions of RAN protocol stack are preferably executed at the radio access as well (e.g., BBU site). Moreover, in order to further reduce latency, UPF could be moved at the BBU site.

mMTCs slice has relaxed latency and data-rate requirements, while the main aspect is the massive number of supported connections. As a consequence, several functions can be placed on NOC and Multi-tenant Cloud platforms. Mobility is usually limited, and data traffic is usually addressed towards a centralized data center on the Cloud. Hence, the functions managing local data traffic and users mobility (i.e., AMF and SMF) are preferably placed on NOC.

Based on previous discussion, the considered values of preferences are reported in Tables 2 to 4 for the different slices. We want to point out, that this is only one of the possible preference configurations, and it is based on previous network/services considerations. Obviously, preferences' values can be different, based also on operators' experience and specific network deployment characteristics. In general, the selected values should fit with the weight and the importance with which we would like to map the function placement on the different layers for different slices.

3. Problem Formulation

Given the sets $\mathcal{F}$ and $\mathcal{S}$, and the network layers constituting the physical infrastructure, the goal of the proposed system is to optimize the function placement while respecting the slices' service requirements and the resource constraints of nodes and communication links. To this aim we model the problem through the definition of a proper cost function to be minimized.

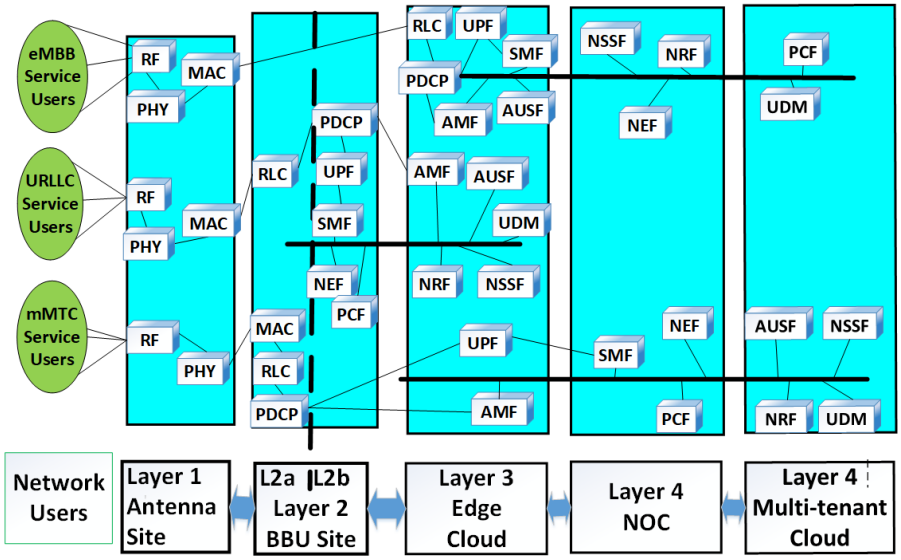

Figure 2: A possible Network Functions Placement.

Table 2: eMBB Preferences.

Function	Layer 1 Antenna site (L1)	Layer 2 BBU site (L2a)	Layer 2 BBU site (L2b)	Layer 3 Edge cloud (L3)	Layer 4 NOC (L4)	Layer 5 Multi-tenant Cloud (L5)
RF	1	0	0	0	0	0
PHY	0.5	0.3	0.2	0	0	0
MAC	0.5	0.3	0.2	0	0	0
RLC	0	0.3	0.2	0.5	0	0
PDCP	0	0.3	0.2	0.5	0	0
UPF	0	0	0.2	0.5	0.3	0
AMF	0	0	0	0.4	0.6	0
AUSF	0	0	0	0.4	0.6	0
SMF	0	0	0	0.4	0.6	0
NSSF	0	0	0	0.6	0.4	0
NEF	0	0	0	0.6	0.4	0
NRF	0	0	0	0	0	1
PCF	0	0	0	0	0	1
UDM	0	0	0	0	0	1

Table 3: URLLC Preferences

Function	Layer 1 Antenna site (L1)	Layer 2 BBU site (L2a)	Layer 2 BBU site (L2b)	Layer 3 Edge cloud (L3)	Layer 4 NOC (L4)	Layer 5 Multi-tenant Cloud (L5)
RF	1	0	0	0	0	0
PHY	0.7	0.2	0.1	0	0	0
MAC	0.7	0.2	0.1	0	0	0
RLC	0.7	0.2	0.1	0	0	0
PDCP	0.2	0.3	0.5	0	0	0
UPF	0	0.1	0.6	0.3	0	0
AMF	0	0	0.1	0.5	0.4	0
AUSF	0	0	0.1	0.5	0.4	0
SMF	0	0	0.1	0.5	0.4	0
NSSF	0	0	0.3	0.2	0.5	0
NEF	0	0	0.3	0.2	0.5	0
NRF	0	0	0	0	0	1
PCF	0	0	0	0	0	1
UDM	0	0	0	0	0	1

Table 4: mMTC Preferences.

Function	Layer 1 Antenna site (L1)	Layer 2 BBU site (L2a)	Layer 2 BBU site (L2b)	Layer 3 Edge cloud (L3)	Layer 4 NOC (L4)	Layer 5 Multi-tenant Cloud (L5)
RF	1	0	0	0	0	0
PHY	0.5	0.4	0.1	0	0	0
MAC	0.5	0.4	0.1	0	0	0
RLC	0.5	0.4	0.1	0	0	0
PDCP	0	0	0.2	0.5	0.3	0
UPF	0	0	0.2	0.3	0.5	0
AMF	0	0	0	0.3	0.7	0
AUSF	0	0	0	0.3	0.7	0
SMF	0	0	0	0.3	0.7	0
NSSF	0	0	0	0	0.6	0.4
NEF	0	0	0	0	0.6	0.4
NRF	0	0	0	0	0	1
PCF	0	0	0	0	0	1
UDM	0	0	0	0	0	1

Since the UP and CP functions are separated, we define $\mathcal{F}_{UP}$ as the set of the UP functions and $\mathcal{F}_{CP}$ as the set of the CP functions, where $\mathcal{F} = \mathcal{F}_{UP} \cup \mathcal{F}_{CP}$, having size F_{UP} and F_{CP}, respectively, with $F = F_{UP} + F_{CP}$. We assume that the computational load to be managed by each function is proportional to the number of active users, and, in the case of the UP functions, also to the amount of traffic (i.e., the information transmitted to/from users), that depends on the application type and, hence, on the slice s. Here we define r_s^u and r_s^c as the UP and CP data generation rates (bits/sec) corresponding to the number of bits generated by each user belonging to sth slice, and transmitted over each plane. Since each slice refers to a specific service, we assume that slices have different r_s^u and r_s^c.

Hence, we can define the cost associated to the placement of a generic function f for the purpose of the slice s as:

$$\eta'(f,s) = U_t(s)\left(u(f)r_s^u + (1-u(f))\,r_s^c\right) \quad (1)$$

where

$$U_t(s) = U_{t_s} \int_{-\infty}^{\infty} x \cdot \mathrm{pdf}_s(x) dx \tag{2}$$

is the average number of *active* users belonging to the slice s, modeled as a function depending on the total number of users of the s-th slice in the considered area, U_{t_s}, and a probability density function, $\mathrm{pdf}_s(\cdot)$, modeling the distribution of the users activity for a given service implemented through the slice s [32]. For modeling the two sets of functions (i.e., UP and CP) we introduce an auxiliary function defined as

$$u(f) = \begin{cases} 1 & \text{if } f \in \mathcal{F}_{UP} \\ 0 & \text{if } f \in \mathcal{F}_{CP} \end{cases}$$

Therefore, the cost of a function is modeled as the amount of data to be processed in the considered time interval. By combining the preference with the execution cost it is possible to define the *preference weighted cost* of a generic function f when placed on a node of the l-th layer for implementing the s-th slice as

$$\eta(f,l,s) = \begin{cases} \frac{1}{p(f,l,s)} \eta'(f,s) & \text{if } p(f,l,s) \neq 0 \\ \text{N/A} & \text{if } p(f,l,s) = 0 \end{cases}$$

that corresponds to set a weighted cost inversely proportional to the preference values, so that highest is the preference lowest is the weight or placing a specific function f of the sth slice on the lth layer. It has to be noticed that in case the preference is 0, the weighted cost is set to N/A that corresponds to have a not allowed placement.

The previous considerations allow us to define the optimal placement problem as:

$$\mathbf{P1} : \min_{\mathbf{A}} \{C(\mathbf{A})\} = \min_{\mathbf{A}} \left\{ \sum_f \sum_s \sum_l a(f,l,s) \eta(f,l,s) \right\} \tag{3}$$

where $C(\mathbf{A}) = \sum_f \sum_s \sum_l a(f,l,s)\eta(f,l,s)$ is the cost function and $\mathbf{A}$ is the function placement three-dimensional matrix whose elements are:

$$a(f,l,s) = \begin{cases} 1 & \text{if the function } f \text{ is placed on the nodes of the layer } l \text{ for the slice } s \\ 0 & \text{otherwise} \end{cases}.$$

P1 is subject to the constraints:

$$\mathbf{C1} : \gamma \sum_s \sum_f a(f,l,s) \eta'(f,s) \leq O_l \tag{4a}$$

$$\mathbf{C2} : \sum_l a(f,l,s) = 1 \tag{4b}$$

$$\mathbf{C3} : \tau_{UP_s} \geq \delta_{UP_s}(\mathbf{A}) + \overbrace{t_0 \sum_f \sum_l a(f,l,s) \eta'(f,s) \frac{1}{g\left(\gamma \sum_p \sum_v a(p,l,v) \eta'(p,v)\right)}}^{\text{processing delay}} \tag{4c}$$

$$\mathbf{C4} : \tau_{CP_s} \geq \delta_{CP_s}(\mathbf{A}) + \overbrace{t_0 \sum_f \sum_l a(f,l,s) \eta'(f,s) \frac{1}{g\left(\gamma \sum_p \sum_v a(p,l,v) \eta'(p,v)\right)}}^{\text{processing delay}}. \tag{4d}$$

The constraint (4a) allows to set a bound on the processing capacity of nodes belonging to the layer l, so that the maximum processing capacity of the layer l is O_l. Here, O_l represents the maximum data processing capacity of the lth layer in terms of Floating Point Operations per Second (FLOPS), while γ is the number of FLOPs required to process a single bit of processing data, supposed to be the same for any function and slice. The constraint (4b) restricts the allocation of a function f when operating in the slice s to one layer l, avoiding to have redundant functions operating for the same slice at different layers. Finally, the constraints (4c) and (4d) allow to set a maximum delay, τ_{UP_s} and τ_{CP_s}, that should be experienced by each communication in the slice s in the UP and CP, respectively. The delay values are composed of two terms:

- $\delta_{UP_s}(\mathbf{A})$ or $\delta_{CP_s}(\mathbf{A})$, corresponding to the latency between the mobile user and the UP end-point or CP end-point, respectively. This term is the sum of propagation delays and transmission delays to go through all links up to the farthest layer, where a function of the slice s is placed;

- the *processing delay* corresponding to the time needed for processing the data related to a given slice. It is modeled as the unit-processing time t_0 (i.e., the time needed for processing one data unit i.e., one bit), multiplied by the processing load (i.e, the whole amount of data produced by the functions of s-th slice in each layer). In order to take into account total processing load at each layer introduced by multiple parallel processing of different functions a decreasing function $g(x)$ is introduced. In general, the processing latency can grow rapidly with additional processing load [33]. Here we assume that up to a given total load $x \leq \mathcal{K} \cdot O_l$, being $\mathcal{K}$ a weighting coefficient, multiple parallel functions can be executed without impacting the operations at layer l, while an additional load introduces a quadratic slow down, according to the model presented

in [33], so that:

$$g(x) = \begin{cases} 1 & \text{if } x \leq \mathcal{K} \cdot O_l \\ \left(\frac{\mathcal{K} \cdot O_l}{x}\right)^2 & \text{otherwise} \end{cases}$$

The above defined problem is an ILP, with non-linear and non-integer constraints. Due to the number of layers, slices and functions, the solutions' space, as later detailed, becomes huge requiring the use of heuristic and meta-heuristic approaches.

4. Proposed Solution

In order to solve the problem in (3), we propose to use an evolutionary based meta-heuristic algorithm that is able to achieve a near-optimal solution respecting the system constraints. In particular, we propose to use a GA. For is characteristics GA is able to generate high-quality solutions in a reduced amount of time, allowing real-time on-demand service function deployment. For comparison purpose we consider also an ESA providing the optimal solution, and two simple heuristic solutions aiming at reducing the computational cost by following the placement preferences. The first, named Maximum Preference Algorithm (MPA) is based on a static 5G deployment where each function of each slice is placed in the layer matching the highest preference irrespective to the other slices and to the network load. This leads to a very simple solution that, however, does not consider the constraints that can be violated. Consequently, we have also considered a Modified Maximum Preference Algorithm (MMPA) that starts for the MPA solution but introduces some limitations to meet the problem constraints. Conversely, the ESA, despite able to find the optimal solutions, is practically unfeasible requiring a huge amount of time for finding a solution.

4.1. Genetic Algorithm (GA)

GAs are evolutionary computation methods that perform a stochastic search based on the principles of natural genetic systems. In general, GA begins with an initial generation with a population space ($\mathcal{PS}$) constituted by random chromosomes ($\mathcal{CV}$). Next, the GA process evolves over several steps through a neighborhood search, where each step involves creating a new $\mathcal{PS}$ with better individuals in it. In the beginning, individual $\mathcal{CV}$ from a current $\mathcal{PS}$ are analyzed through a fitness function $\mathcal{FS}$ and, through their fitness values, a set of chromosomes is selected for the creation of the next-generation $\mathcal{PS}$. The formation of the next-generation $\mathcal{CV}$ set is based on the transfer of the individuals with better fitness values from a current $\mathcal{PS}$ to the next $\mathcal{PS}$, developing new individuals through the mutation and crossover processes. In the mutation process, a population point is formed through random changes in the selected solution, while, in the crossover process, two chromosome set constitute a $\mathcal{CV}$ for the next generation by combining their parts with better fitness values. Each evaluation creates a better solution set and finally ends up providing a solution point with a higher fitness value. The individual elements of the GA process are provided below.

4.1.1. Chromosome Coding

In this work, each network function placement variable $a(f, l, s) = \{0, 1\}$ is considered as a gene in the GA. All the genes together compose one chromosome $\mathcal{CV}$. Thus, each $\mathcal{CV}$ is constituted by $F \times L \times S$ genes, which take values 0 or 1. A representation of a $\mathcal{CV}$ is given in Figure 3.

Figure 3: Chromosome Coding

4.1.2. Initial Population

In this paper, the initial population $\mathcal{PS}$ is defined as a set of 100 possible feasible solutions for the network function placement problem defined in (3). GAs work by generating multiple possible solutions, ranking them in terms of the objective functions and generating new solutions based on a combination of the best, selected at the previous stage [34, 35]. In order to do this we need multiple solutions since the first step. In the proposed procedure we suppose to start from 100 solutions. All of them should be feasible in order to have a starting point that makes sense. Each placement matrix $\mathbf{A}$ is reshaped as a $\mathcal{CV}$ array having size $1 \times (F \times L \times S)$. Among the 100 starting population, the first two chromosomes are the solutions of MPA and MMPA methods, i.e., $\mathbf{A}_{MPA}$ and $\mathbf{A}_{MMPA}$, while the remaining 98 points are generated by changing an arbitrary number of genes from the previous two chromosomes. Since MPA and MMPA methods provide solutions potentially closer to the optimal points, using their solutions as an initial point in the $\mathcal{PS}$ increases the efficiency of GA.

4.1.3. Fitness Function

The fitness function $\mathcal{FS}$ in GA is a function that measures the performance of each $\mathcal{CV}$ for the considered problem and assigns fitness scores to them. In this work, the fitness of each point in $\mathcal{PS}$ is measured through the preference weighted execution cost function over all f, l, and s. Thus,

$$\mathcal{FS}(\mathcal{CV}) = C(\mathbf{A}(\mathcal{CV}))$$

where, $\mathbf{A}(\mathcal{CV})$ is the solution $\mathbf{A}$ constructed by rearranging $\mathcal{CV}$. The set of constraints defined in (4a) to (4d) are also considered during the evaluation of each $\mathcal{CV}$.

4.1.4. Neighbourhood Search and GA Operators

In GA, neighborhood search corresponds to find a better $\mathcal{CV}$ set for the next generation of the algorithm. It includes the measurement of fitness values of each individuals in the current population using $\mathcal{FS}$, convert raw fitness scores $\mathcal{FS}(\mathcal{CV})$ into an expected range of values, selecting parents for the next generation through selection function, and reproducing new $\mathcal{CV}$s through elitism, mutation, and crossover operations. The raw fitness scores are evaluated through a Fitness Scaling Function (F_f) that scales the raw fitness scores ($\mathcal{FS}(\mathcal{CV})$) into values suitable for the selection function (S_f). The elite operation allows a fraction of current generation $\mathcal{CV}$s with better

Table 5: GA Operator Functions and Parameters

Population Space Size ($\mathcal{PS}$)	100
Elite Count (e)	$0.05 \times \mathcal{PS}$
Crossover Count (Cc)	$0.9 \times \mathcal{PS}$
Selection Function (S_f)	Uniform Selection
Crossover Function (C_f)	Random Binary Vector based Crossover
Mutation Function (M_f)	Adaptive Mutation based on the Previous Results
Fitness Scaling Function (F_f)	$1/\sqrt{rank(\mathcal{FS}(\mathcal{CV}))}$

fitness values directly added to the next generation. The total number of elite $\mathcal{CV}$s passed into the next generation are based upon the elite count e. In general, mutation operation involves creating new chromosomes with better fitness through random changes in the selected chromosome. The mutation function (M_f) allows a selection of genes for the mutation operations. In crossover operation, two chromosome vectors form a new better fit solution point for the next generation by combining their parts. The crossover operation is done through the crossover function (C_f), which defines the policy for the crossover operation. With the help of these operators, GA evolves over several generations and results in a better fit solution for the problem at hand. We have used a Matlab simulation environment for the implementation of GA [35]. The main parameters and GA operator functions used during the simulation are given in the Table 5.

Algorithm 1 lists the main steps used in the GA process. Initially, $\mathcal{PS}$ is generated from MPA and MMPA results (lines 1-2). $\mathcal{PS}$ evaluation and scaling is performed in lines 5 to 8. The selection function (S_f) has been used to select better $\mathcal{CV}$s based upon their fitness (Line 9-11). While in Line 12 to 15, a new set of individual $\mathcal{CV}$s is generated by using GA operators and used for the next generation of GA. The above procedure is repeated until predefined stopping criteria are reached. In this work, we considered as stopping criteria the maximum number of GA iterations (G_{max}) and two different tolerance values. In particular, we considered that the process stops if, for a given number of consecutive iterations (G^{fl}_{max}), the fitness function remains within a given tolerance ($\mathcal{F}l$) and the constraint feasibility (Cl) is respected. In this work $G_{max} = 1000$, $G^{fl}_{max} = 100$, $\mathcal{F}l = 10^{-6}$ and $Cl = 10^{-6}$ are the numerical values of the stopping criteria and the GA process stops if any one of this condition is satisfied.

Algorithm 1 Genetic Algorithm

Input: $\mathcal{FS}, e, M_f, C_f, G_{max}, A_{MPA}, A_{MMPA}$
Output: Placement matrix $\mathbf{A}_{GA}$
1: Read A_{MPA} and A_{MMPA}
2: Generate the initial population $\mathcal{PS}$ by randomly changing the chromosome of $\mathbf{A}_{MPA}$ and $\mathbf{A}_{MMPA}$
3: **while** *stopping_criteria* is false **do**
4: **function** EVALUATE($\mathcal{PS}$)
5: Find $\mathcal{FS}(\mathcal{CV})$, $\forall \mathcal{CV} \in \mathcal{PS}$
6: Convert $\mathcal{FS}(\mathcal{CV})$ using F_f.
7: **end function**
8: **function** SEARCH($\mathcal{PS}$)
9: Select better fit individuals using S_f
10: **end function**
11: **function** CREATE($\mathcal{PS}$)
12: Perform Elite, Mutation and Crossover operations (using e, M_f, C_f) for generating new $\mathcal{CV}$s.
13: **end function**
14: Replace current $\mathcal{PS}$ with new set of $\mathcal{CV}$s.
15: **end while**
16: **return** Placement matrix $\mathbf{A}_{GA}$

4.2. Benchmark Methods

In this work, we have considered three different benchmark methods for comparing the performance of GA.

4.2.1. Maximum Preference Algorithm (MPA)

The goal of the MPA heuristic is to maximize the operator needs by placing the network functions on the layer with the highest preference value. Therefore,

$$a(f,l,s) = \begin{cases} 1 & \text{if } p(f,l,s) = \max\limits_{l=1,\cdots,L} p(f,l,s) \\ 0 & \text{otherwise} \end{cases} \quad \forall f \in \mathcal{F} \text{ and } s \in \mathcal{S} \quad (5)$$

The solution is very simple, but since the placement is fixed, i.e., in the layer that has the maximum preference (different for each function of each slice), network's constraints, slices' requirements as well as cross-slice effects are not considered. As a consequence MPA is not always able to achieve a feasible solution, respecting all the constraints, as it will be shown. In particular, we want to underline that the MPA can be seen as a particular case of the proposed approach when the preferences are set to 1 only for one layer and zero for the others for each function of each slice.

4.2.2. Modified Maximum Preference Algorithm (MMPA)

In order to achieve a feasible solution (i.e., respecting the network/services constraints) maintaining a low implementation cost we consider also a modification of the MPA solution. MMPA performs a one-dimensional search for each network function and places them on the layer with the highest preference provided that all constraints are satisfied.

In particular, starting from the evaluation of the delay bounds in (4c) and (4d), the algorithm is initialized by reducing the possible initial layers to those able to respect the delay constraints; to this aim we consider the worst-case latency condition between one user and the user/control plane end-points. By defining $\varphi(l)$ as the propagation and transmission delay between $(l-1)$-th and l-th layer, we can set L_{UPs} and L_{CPs} as:

$$L_{UPs} \rightarrow \max\left\{ l : \sum_{l} \varphi(l) \leq \tau_{UP_s} \right\} \quad (6a)$$

$$L_{CPs} \rightarrow \max\left\{ l : \sum_{l} \varphi(l) \leq \tau_{CP_s} \right\} \quad (6b)$$

Following this, the algorithm finds the layer with the highest preference value to host a particular network function while respecting the delay constraints. In case the layer with the highest preference is not able to fulfill the computational capacity constraint for hosting the allocated network function, we suppose that such function is allocated to the layer having the immediately lower preference value; the process continues until the algorithm is able to find a proper place for each network function, or there are not more layers where to place the function.

Thus, we have:

$$a(f,l,s) = \begin{cases} 1 & \text{if } \eta'(f,s) \leq \bar{O}_l \quad \wedge \\ & \qquad p(f,l,s) = \max\limits_{l=1,\ldots,L_{UP_s}/L_{CP_s}} \{p(f,l,s)\} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall s \in \mathcal{S}, \forall f \in \mathcal{F}_{UP}/\mathcal{F}_{CP} \quad (7)$$

The processing capacity of each layer is updated once a particular network function is placed on it. The updated value of the processing capacity, defined as $\bar{O}_l$, is given by,

$$\bar{O}_l \leftarrow \bar{O}_l - \eta'(f,s) \quad \forall l$$

where $\eta'(f,s)$ is the processing cost of the function f of slice s. This process is repeated for all functions from different slices. Generally, UP functions have more stringent latency requirements than CP functions; therefore, in the function placement process we decide to give a higher priority to the UP functions by placing them first.

Despite MMPA tries to satisfy the constraints, its flexibility is limited and, as we can see later in the numerical results, constraints are not always respected.

4.2.3. Exhaustive Search Algorithm (ESA)

The ES method can be used to find the optimal solution through a sequential search by examining the space $\mathcal{SP} = \{\mathbf{A}\}$ containing all possible solutions ($\mathbf{A}$) available for the network function placement problem defined in (3). In the end, ESA results in an optimal network function placement matrix ($\mathbf{A}_{ESA}$) providing the minimum cost, while satisfying all constraints defined in (4a) to (4d).

Given the complex nature of the system, the solution space contains an enormous number of possible solutions

$$\prod_{s=1}^{S} \prod_{f=1}^{F} \kappa_{f,s}$$

where,

$$\kappa_{f,s} = \left\{ \sum_{l=1}^{L} l \mid p(f,l,s) \neq 0 \right\} \qquad \forall f, s$$

is a mathematical operator measuring the number of layers with nonzero preference values for each network function from all slices. ESA requires a high computational complexity and can be used only for a limited amount of scenarios and it is not a practical solution from the implementation point of view. However, it is useful to evaluate the accuracy of the proposed GA algorithm.

4.3. Comparison of different methods

In Table 6 the algorithms previously introduced are compared. In general, MPA and MMPA are computationally inexpensive and fast techniques, but lack the search flexibility and thus can fail in meeting the service/network constraints thus producing unpractical solutions. ESA is able to find the optimal solution but requiring a huge amount of time. Finally, GA achieves a good trade-off among the previous approaches. It is able to find a practical solution with a reasonable computational complexity. The computation complexity for MPA and MMPA grows linearly with inputs. i.e., F, L, and S. On the other hand, ESA needs to explore all possible solutions for finding the global optimal, and its complexity grows exponentially with respect to F, S, and L. With the presence of stochastic operators, it's difficult to provide numerical expression for the GA complexity, so we resort to the simulation time required for the GA operations. To this aim, in the last column of the Table 6, we record the time required to simulate one solution point in Matlab for the considered algorithms. Moreover, it has to be noticed that while GA, MPA and MMPA have been tested over a commercial computer equipped with Intel(R) Core i5-8250U CPU @ 1.60GHz, the ESA has to be tested on a rack workstation equipped with eight dual core Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, highlighting the complexity of the ESA w.r.t the other approaches.

Table 6: Overall Comparison among Algorithms.

Algorithm	**Computational Complexity**	**Search Flexibility**	**Type**	**Simulation Time per solution point [sec]**
MPA	$O(F \cdot L \cdot S)$	Very Low	Heuristic	0.089
MMPA	$O(F \cdot L \cdot S)$	Low	Heuristic	0.1085
GA	-	Good	Metaheuristic	24.03
ESA	$O\left(L^{F \cdot S}\right)$	Complete Search	Optimal	824.82

5. Numerical Results

In this section numerical results obtained through computer simulations are given, aiming at analysing and comparing the performance of the considered solutions.

In Table 7 the most important parameters are listed. Each layer is supposed to have a limited processing capability and can process up to O_l FLOPS, while $\varphi(l)$ specifies the sum of propagation and transmission delays between different layers of the system architecture [8]. These values are based on different transmission mediums as specified before.

Table 7: Simulation Parameters.

Layer Processing Capacity $(O_l)[FLOPS]$	$(0.75, 1.25, 1.25, 3, 4.5, 5) \times 10^{15}$
$\varphi(l)$ [msec]	UE $\overset{0.5}{\longrightarrow} L1 \overset{1}{\rightarrow} L2a \overset{3}{\rightarrow} L2b \overset{11.5}{\longrightarrow} L3 \overset{20}{\rightarrow} L4 \overset{40}{\longrightarrow} L5$
t_0 [msec]	1.5×10^{-13}
$\mathcal{K}$	0.33
γ [FLOPs]	10^3

In order to check the effectiveness of the proposed model, an extensive set of Matlab based simulations is performed by considering different use cases. The Matlab scripts have been released and made are available to the general public for testing purposes on the GitHub platform[1]. In each use case, the

[1] https://github.com/swapnilshinde2/5G-NFV-Slice-preferences

key parameters of a particular service have been changed while maintaining other parameters fixed. Basic key parameters are listed in Table 8.

Table 8: Use Case I: Input Parameters

Parameters	**eMBB**	**URLLC**	**mMTC**
User-Plane Data Rate (r_s^u) [Mb/sec]	1500	100	0.5
Users	2000	1000	1000000
Average per slice users activity (2)	0.3	0.2	0.1
τ_{UPs} [msec]	50	5	1000
τ_{CPs} [msec]	100	100	100

The table reports the number of users requesting each service, the average users activity for each slice as introduced in (2), and the data rate requested by the service. The CP is supposed to handle a very few packets, and, therefore, we consider that r_s^c is 1 Mb/sec for all slices. The remaining parameters τ_{UPs} and τ_{CPs} represent the UP and CP latency constraints. The average number of active users users belonging to the particular slice s can be determined by using the mean value of the user activity distribution (K_{t_s}) as $U_t(s) = U_{t_s} K_{t_s}$.

5.1. Use Case I: eMBB slice data rate variation

In general, eMBB services are characterized by high data rates while having moderate latency requirements. Hence, in Case I, we suppose to vary the data rate of the users requiring the eMBB slice while keeping constant all other parameters. In particular UP data rate values for the eMBB slice change in the range [200 Mb/s ÷ 2 Gb/s].

Figures 4-10 report the performance of the four algorithms previously introduced. In particular, in Figure 4, the cost value as defined in (3) is plotted as a function of the eMBB user data rate. As shown, by increasing the data rate, the algorithms have different behaviours. MPA technique is able to provide a lower bound on the cost function but, as shown in following figures, fails to satisfy both the layer processing capacity and the delay constraints. MMPA method is able to adjust its network function placement according to the processing load limits of each layer, but its lack of flexibility in distributing the load on different layers ends up adding more processing delay, thus delay constraints are not satisfied. This is evident in Figures 5 and 6, where MPA and MMPA fail to satisfy the UP requirements of URLLC and eMBB slices. Though GA has a slight increase in terms of cost for higher data rates, it adapts its network function placement according to latency constraints and computational capacity of each layer. Figure 7 shows the computational load generated in each layer by each solution technique. ESA provides the global optimal solution and can be seen as a benchmark approach to compare the performance of other techniques. However, it requires exploring the complete solution space for finding the optimal solution. Here for the considered system model, the total number of points is huge ($\approx 3^{26}$), and practically unfeasible to be solved in Matlab. Therefore, to reduce the complexity of the ESA simulation, we have used a function grouping-based approach in which all the functions having similar preferences are considered as a single functional entity and placed on the same layer (e.g., PHY, MAC, and RLC functions of URLLC slice have similar preferences and so treated as a one function entity during simulation of ESA).

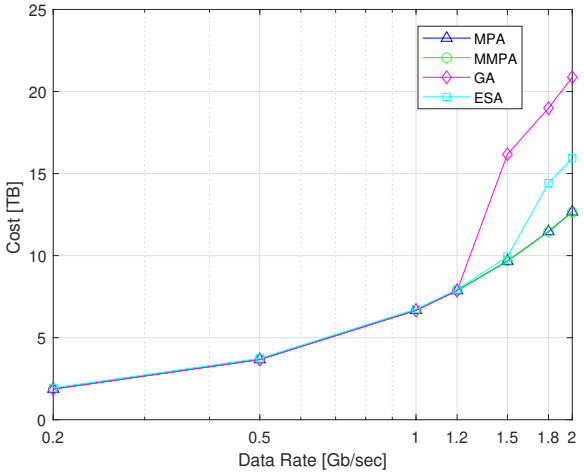

Figure 4: Cost Function when varying the data rate of the eMBB users

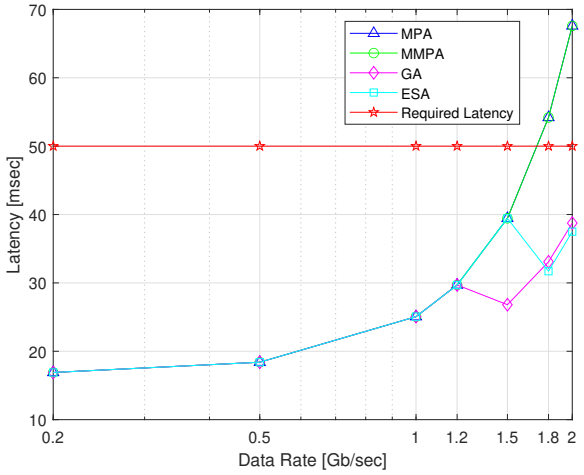

Figure 5: eMBB Latency when varying the data rate of the eMBB users

In order to show how functions are moved from one network element to another when system parameters change we use the network function placement graph. This is represented in Figure 8 as an example for explaining it. The horizontal axis lists the available layers, while each row on the vertical axis corresponds to the network functions. The colored blocks indicate the placement of the function on a specific layer.

The Figures 9 and 10 show the network function placement generated by GA and ESA for different data rate values of eMBB slice. It is possible to note that as the data rate increases, not only the eMBB slice changes its functions placement, but also URLLC and mMTC slices have many changes despite their parameters are constant. This is a consequence of the cross-slice effect. It is possible to notice that the functions of different slices move from their preferred location to less pre-

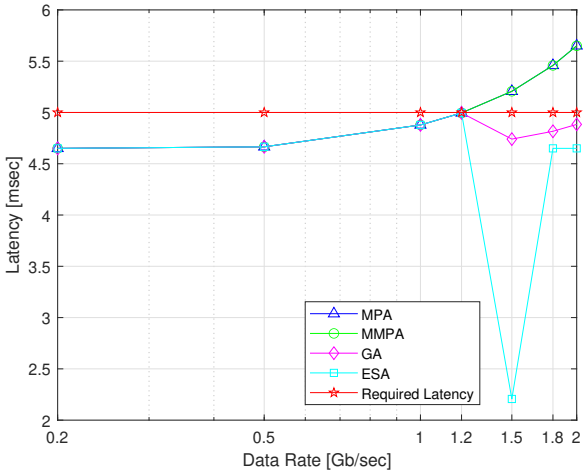

Figure 6: URLLC Latency when varying the data rate of the eMBB users

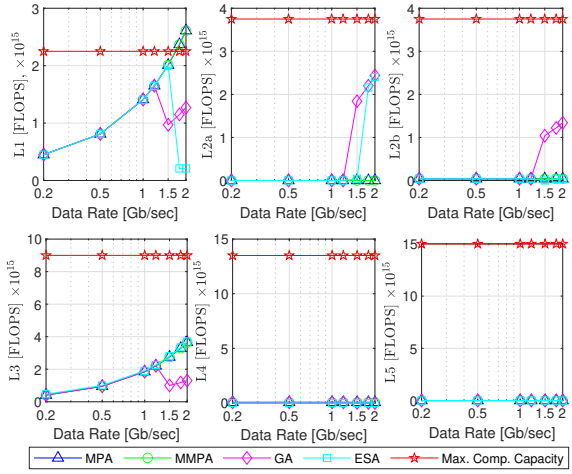

Figure 7: Computational Capacity vs Computation Load when varying the data rate of the eMBB users

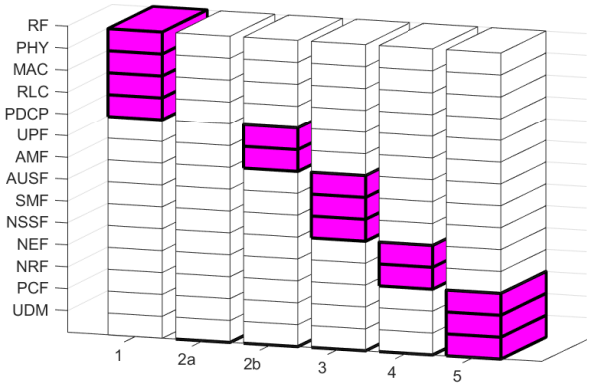

Figure 8: Network Function Placement.

ferred ones as the data rate increases. In particular, up to 1200 Mb/sec, several UP functions from different slices are packed on L1 while L2a is not used. When the data rate increases, the processing delay increases for the higher traffic load to be managed, hence the functions must be moved in L2a. For example observing the URLLC slice we can see that up to 1200Mb/sec PDCP and UPF functions are placed on layer L2b, because the L1 processing latency is low. However, as the data rate and hence, the data amount to be processed increases also the processing latency of L1 increases. As a consequence the URLLC slice has to move PDCP and UPF functions on layer L2a thus compensating the increase in processing latency of L1 with a reduced transmission delay for satisfying the 5 msec latency constraint. At 200Mb/s the two functions are again moved on L2b because the L1 has been lightened of some functions of the eMBB slice. Due to this, the overall UP latency of the URLLC slice decreases sharply (Figure 6), however, this results impacts in an increment of the cost function value which can be seen from Figure 4. ESA distributes the computation load from slices over different layers better than the GA. Indeed, GA makes several changes in the function placement and eventually converges to the local optimal solution satisfying all latency constraints with a higher cost. However, we can observe that the overall behaviour is similar.

5.2. Use Case II: variable user plane latency of the URLLC slice

In the Use Case II, the UP latency requirement is changed in the URLLC slice, allowing to understand the impact of the latency on the proposed placement algorithms. In particular, we vary the UP latency requirements from 2 msec up to 25 msec. Figures 11-14 show the performance of the different algorithms when URLLC latency requirement is changed.

Figure 11 shows the cost function values for the considered algorithms. Though cost values for the ESA and GA techniques are slightly higher for some latency values respect to the two heuristics, they can perform the network function placement satisfying the latency requirements, while MPA and MMPA fail as evident in Figure 12. For the case of 5 msec latency, GA converges to the local optimal solution that follows all constraints while ESA searches through the entire solution space for finding the optimal solution. Therefore there is a difference in the cost function value of ESA and GA.

Figures 13 and 14 show the network function placement solutions for GA and ESA methods. It is possible to notice that for URLLC services with extremely low latency, network functions are deployed much closer to the end-users, likely on L1 and L2, while when latency conditions become non-critical, functions can be deployed more smoothly on the higher layers of the architecture.

5.3. Use Case III: variable number of users in the mMTC slice

In case of mMTC services, the number of devices to be connected is supposed to be huge. Consequently, here we considered the performance of different methods by varying the number of mMTC user connections from 100000 up to 10000000.

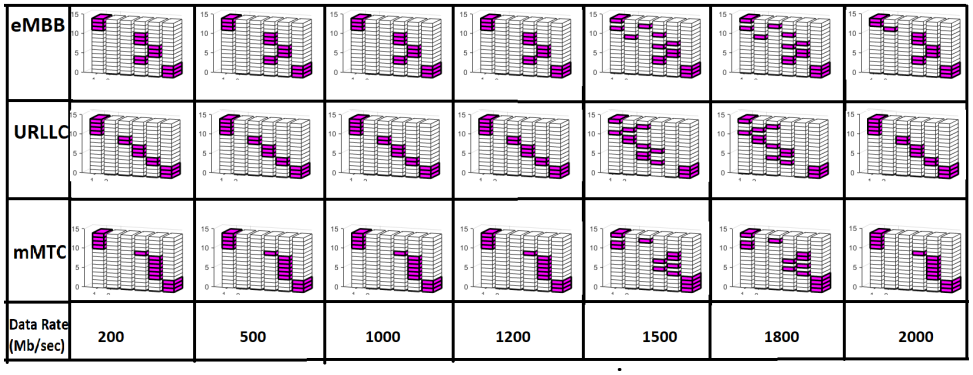

Figure 9: Network Function Placement when varying the data rate of the eMBB users for the GA.

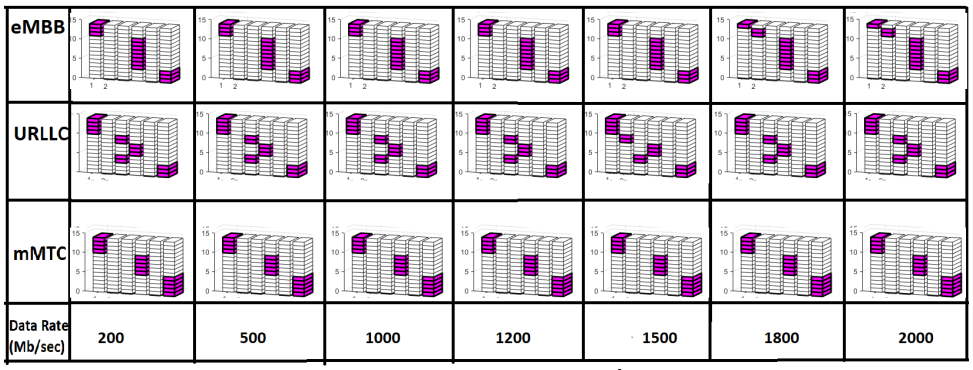

Figure 10: Network Function Placement when varying the data rate of the eMBB users for the ESA.

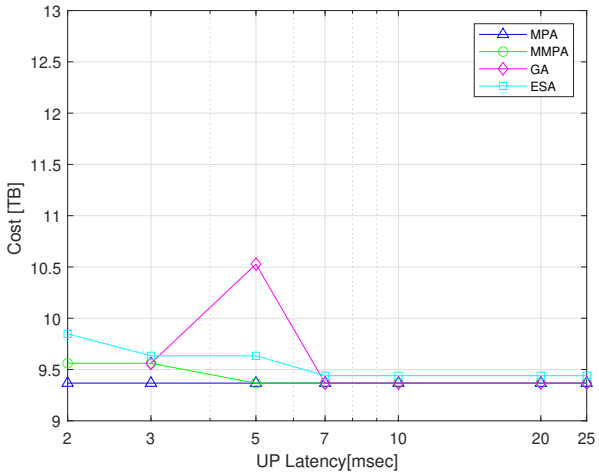

Figure 11: Cost Function when varying the UP latency requirement of URLLC users

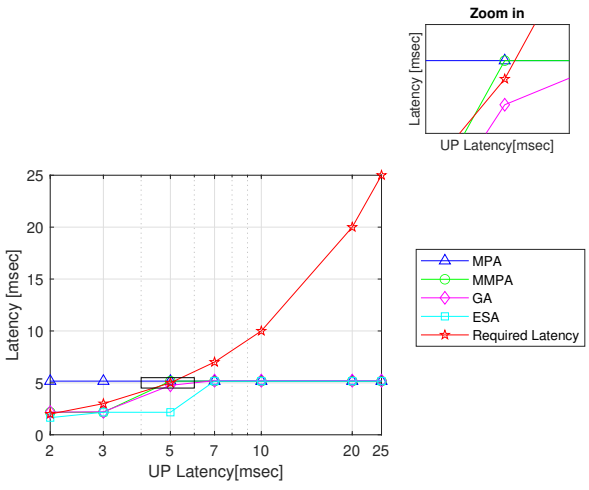

Figure 12: URLLC Latency when varying the UP latency requirement of URLLC users

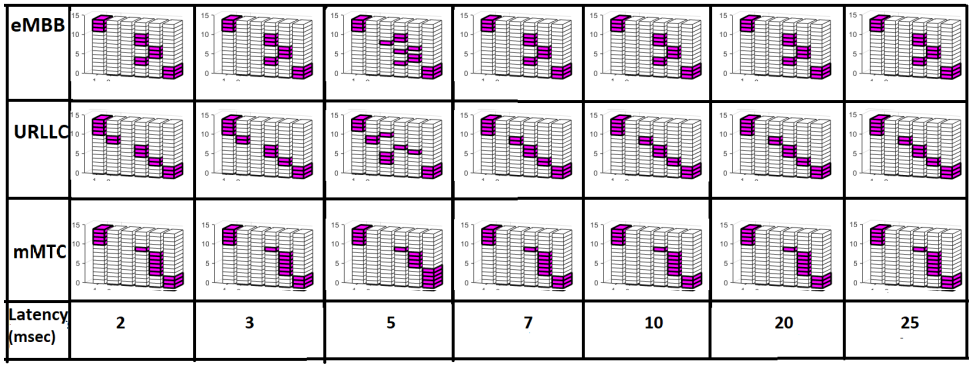

Figure 13: Network Function Placement when varying the UP latency requirement of URLLC users for the GA

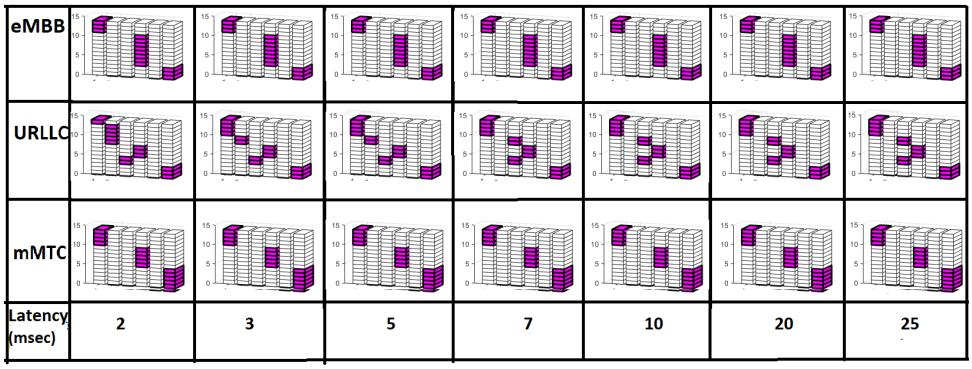

Figure 14: Network Function Placement when varying the UP latency requirement of URLLC users for the ESA.

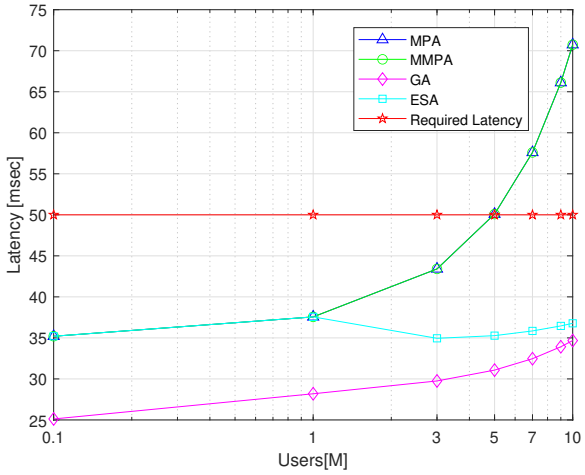

Figure 15: eMBB Latency when varying the number of mMTC users

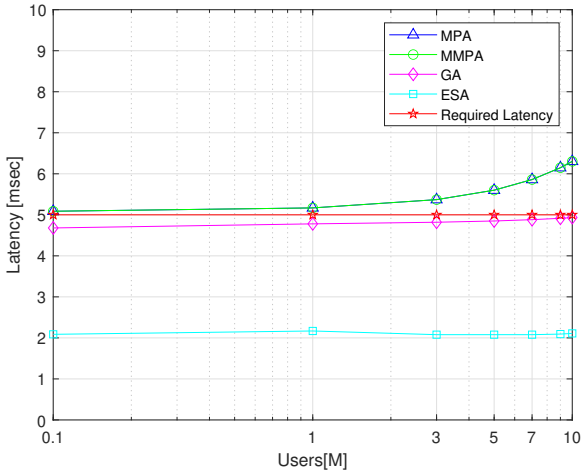

Figure 16: URLLC Latency when varying the number of mMTC users

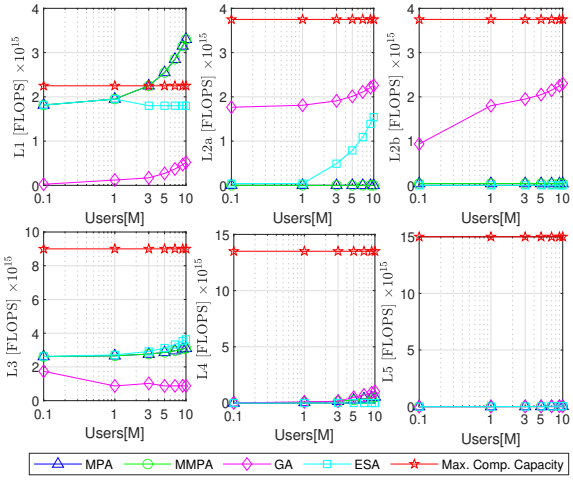

Figure 17: Processing Capacity vs Computation Load when varying the number of mMTC users

The main results for this case are presented in Figures 15, 16 and 17. In particular, Figures 15 and 16 show the performance of different schemes in terms of the eMBB and URLLC latency, respectively. With such a high number of user connection requests from mMTC services, MPA and MMPA techniques fail to keep UP latency values for these services under control mainly because of inadequate search flexibility. On the other hand, GA has better performance and is able to satisfy all the requirements. It should also be noted that though GA and ESA techniques latency values are changing slowly for the URLLC case, while a significant change can be seen in the eMBB slice. Figure 17 presents the processing load generated by each technique over different layers and shows that GA can adapt its network function placement based upon the available layer capacity.

6. Conclusion

This paper addressed the network function placement problem in a multi-service 5G network architecture. The problem has been modeled through a proper objective function that takes into account the computational load of each function together with a preference parameter that allows to bias the correct placement of the functions for each service slice. Moreover, constraints on service requirements and network resources have been considered.

The proposed solution is based on an evolutionary-based meta-heuristic algorithm. Moreover, the optimal solution achieved by means an exhaustive search and two simple heuristics have been considered as benchmarks. Numerical results show that the proposed solution allows to achieve a near-optimum solution able to satisfy the imposed constraints even when these are very stringent and with a reasonable complexity. Conversely, the two heuristics fails in meeting the constraints when these become stringent.

The analysis of the function placement shows that the GA method allows flexibility, thus the functions can be moved depending on the scenario and that a tight correlation among the slices is achieved.

As a final comment it is worth to be noticed that the provided solution can deserve as a starting point for training a Machine learning based algorithm. As an example, the basic configurations obtained by means of our approach, and based on averaged values, could be used for training a neural network that could be used for providing real-time solutions based on instantaneous values. This could be considered in an evolution of our work.

References

[1] Ericsson mobility report, Tech. rep., Ericsson, Stockholm, Sweden (Jun. 2021).
URL https://www.ericsson.com/4a03c2/assets/local/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, H. Flinck, Network slicing and softwarization: A survey on principles, enabling technologies, and solutions, IEEE Communications Surveys & Tutorials 20 (3) (2018) 2429–2453. doi:10.1109/COMST.2018.2815638.

[3] S. Abdelwahab, B. Hamdaoui, M. Guizani, T. Znati, Network function virtualization in 5G, IEEE Communications Magazine 54 (4) (2016) 84–91. doi:10.1109/MCOM.2016.7452271.

[4] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, L. Dittmann, Cloud RAN for mobile networks—a technology overview, IEEE Communications Surveys & Tutorials 17 (1) (2015) 405–426. doi:10.1109/COMST.2014.2355255.

[5] M. C. Filippou, D. Sabella, M. Emara, S. Prabhakaran, Y. Shi, B. Bian, A. Rao, Multi-access edge computing: A comparative analysis of 5G system deployments and service consumption locality variants, IEEE Communications Standards Magazine 4 (2) (2020) 32–39. doi:10.1109/MCOMSTD.001.1900034.

[6] D. Milojicic, The edge-to-cloud continuum, Computer 53 (11) (2020) 16–25. doi:10.1109/MC.2020.3007297.

[7] F. van Lingen, M. Yannuzzi, A. Jain, R. Irons-Mclean, O. Lluch, D. Carrera, J. L. Perez, A. Gutierrez, D. Montero, J. Marti, R. Maso, J. P. Rodriguez, The unavoidable convergence of NFV, 5G, and fog: A model-driven approach to bridge cloud and edge, IEEE Communications Magazine 55 (8) (2017) 28–35. doi:10.1109/MCOM.2017.1600907.

[8] S. E. Elayoubi, S. B. Jemaa, Z. Altman, A. Galindo-Serrano, 5G RAN slicing for verticals: Enablers and challenges, IEEE Communications Magazine 57 (1) (2019) 28–34. doi:10.1109/MCOM.2018.1701319.

[9] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, R. Du, End-to-end network slicing in radio access network, transport network and core network domains, IEEE Access 8 (2020) 29525–29537. doi:10.1109/ACCESS.2020.2972105.

[10] Y. L. Lee, J. Loo, T. C. Chuah, L.-C. Wang, Dynamic network slicing for multitenant heterogeneous cloud radio access networks, IEEE Transactions on Wireless Communications 17 (4) (2018) 2146–2161. doi:10.1109/TWC.2017.2789294.

[11] P. Zhao, H. Tian, S. Fan, A. Paulraj, Information prediction and dynamic programming-based RAN slicing for mobile edge computing, IEEE Wireless Communications Letters 7 (4) (2018) 614–617. doi:10.1109/LWC.2018.2802522.

[12] H. Halabian, Distributed resource allocation optimization in 5G virtualized networks, IEEE Journal on Selected Areas in Communications 37 (3) (2019) 627–642. doi:10.1109/JSAC.2019.2894305.

[13] L. Liang, Y. Wu, G. Feng, X. Jian, Y. Jia, Online auction-based resource allocation for service-oriented network slicing, IEEE Transactions on Vehicular Technology 68 (8) (2019) 8063–8074. doi:10.1109/TVT.2019.2924456.

[14] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, Z. Zhu, Resource allocation for network slicing in 5G telecommunication net-

works: A survey of principles and models, IEEE Network 33 (6) (2019) 172–179. doi:10.1109/MNET.2019.1900024.

[15] C. Pham, N. H. Tran, S. Ren, W. Saad, C. S. Hong, Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach, IEEE Transactions on Services Computing 13 (1) (2020) 172–185. doi:10.1109/TSC.2017.2671867.

[16] M. Chen, Y. Sun, H. Hu, L. Tang, B. Fan, Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling, IEEE Transactions on Green Communications and Networking 5 (1) (2021) 29–40. doi:10.1109/TGCN.2020.3042675.

[17] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, X. Fu, Delay-aware virtual network function placement and routing in edge clouds, IEEE Transactions on Mobile Computing 20 (2) (2021) 445–459. doi:10.1109/TMC.2019.2942306.

[18] A. Laghrissi, T. Taleb, A survey on the placement of virtual resources and virtual network functions, IEEE Communications Surveys & Tutorials 21 (2) (2019) 1409–1434. doi:10.1109/COMST.2018.2884835.

[19] A. De Domenico, Y.-F. Liu, W. Yu, Optimal virtual network function deployment for 5G network slicing in a hybrid cloud infrastructure, IEEE Transactions on Wireless Communications 19 (12) (2020) 7942–7956. doi:10.1109/TWC.2020.3017628.

[20] D. Bhamare, A. Erbad, R. Jain, M. Zolanvari, M. Samaka, Efficient virtual network function placement strategies for cloud radio access networks, Computer Communications 127 (2018) 50–60. doi:10.1016/j.comcom.2018.05.004.

[21] S. T. Arzo, R. Bassoli, F. Granelli, F. H. P. Fitzek, Study of virtual network function placement in 5G cloud radio access network, IEEE Transactions on Network and Service Management 17 (4) (2020) 2242–2259. doi:10.1109/TNSM.2020.3020390.

[22] T. Truong-Huu, P. Murali Mohan, M. Gurusamy, Service chain embedding for diversified 5G slices with virtual network function sharing, IEEE Communications Letters 23 (5) (2019) 826–829. doi:10.1109/LCOMM.2019.2900888.

[23] D. Li, P. Hong, W. Wang, J. Pei, Virtual network function placement with function decomposition for virtual network slice, in: 2018 IEEE Conference on Standards for Communications and Networking (CSCN), Paris, France, 2018, pp. 1–4. doi:10.1109/CSCN.2018.8581851.

[24] N. Promwongsa, M. Abu-Lebdeh, S. Kianpisheh, F. Belqasmi, R. H. Glitho, H. Elbiaze, N. Crespi, O. Alfandi, Ensuring reliability and low cost when using a parallel VNF processing approach to embed delay-constrained slices, IEEE Transactions on Network and Service Management 17 (4) (2020) 2226–2241. doi:10.1109/TNSM.2020.3029108.

[25] S. D. A. Shah, M. A. Gregory, S. Li, Cloud-native network slicing using software defined networking based multi-access edge computing: A survey, IEEE Access 9 (2021) 10903–10924. doi:10.1109/ACCESS.2021.3050155.

[26] A. Alabbasi, X. Wang, C. Cavdar, Optimal processing allocation to minimize energy and bandwidth consumption in hybrid CRAN, IEEE Transactions on Green Communications and Networking 2 (2) (2018) 545–555. doi:10.1109/TGCN.2018.2802419.

[27] S. Redana, O. Bulakci, C. Mannweiler, L. Gallo, A. Kousaridas, D. Navrátil, A. Tzanakaki, J. Gutiérrez, H. Karl, P. Hasselmeyer, A. Gavras, S. Parker, E. Mutafungwa, 5G PPP Architecture Working Group - View on 5G Architecture, Tech. rep., Zenodo, version 3.0 (2019). doi:10.5281/zenodo.3265031.

[28] Y. Tsukamoto, R. K. Saha, S. Nanba, K. Nishimura, Experimental evaluation of RAN slicing architecture with flexibly located functional components of base station according to diverse 5G services, IEEE Access 7 (2019) 76470–76479. doi:10.1109/ACCESS.2019.2922251.

[29] 5G NR logical architecture and its functional splits, White paper, Parallel Wireless.
URL `https://www.parallelwireless.com/white-papers/5g-functional-splits/`

[30] IMT vision - Framework and overall objectives of the future development of IMT for 2020 and beyond, Rec. M.2083-0, ITU-R (Sep. 2015).

[31] 5G white paper, Tech. rep., Next Generation Mobile Networks Alliance (2015).
URL `https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf`

[32] P. Popovski, K. F. Trillingsgaard, O. Simeone, G. Durisi, 5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view, IEEE Access 6 (2018) 55765–55779. doi:10.1109/ACCESS.2018.2872781.

[33] G. Li, J. Wang, J. Wu, J. Song, Data processing delay optimization in mobile edge computing, Wireless Communications and Mobile Computing (2018). doi:10.1155/2018/6897523.

[34] A. Eiben, J. Smith, Introduction to Evolutionary Computing, 2nd Edition, Natural Computing Series, Springer, Berlin, Heidelberg, 2015. doi:10.1007/978-3-662-44874-8.

[35] A. Chipperfield, P. Fleming, The MATLAB genetic algorithm toolbox, in: IEE Colloquium on Applied Control Techniques Using MATLAB, London, UK, 1995, pp. 10/1–10/4. doi:10.1049/ic:19950061.