Multi-site Resource Allocation in a QoS-Aware 5G Infrastructure

Raffaele Bolla, *Senior Member, IEEE*, Roberto Bruschi, *Senior Member, IEEE*, Franco Davoli, *Life Senior Member, IEEE*, Chiara Lombardo and Jane Frances Pajo

***Abstract*— Network softwarization has paved the way for 5G technologies, and a wide-range of (radically new) verticals. As the telecommunications infrastructure evolves into a sort of distributed datacenter, multiple tenants such as vertical industries and network service providers share its aggregate pool of resources (e.g., networking, computing, etc.) in a layered "as-a-Service" approach exposed as slice abstractions. The challenge remains in the coordination of various stakeholders' assets in realizing end-to-end network slices and supporting the multi-site deployment and chaining of the micro-service components needed to implement cloud-native vertical applications (vApps). In this context, particular care must be taken to ensure that the required resources are identified, made available and managed in a way that satisfies the vApp requirements, allows for a fair share of resources and has a reasonable impact on the overall vApp deployment time. With these challenges in mind, this paper presents the Resource Selection Optimizer (RSO) – a software-service in the MATILDA Operations Support System (OSS), whose main goal is to select the most appropriate network and computing resources (according to some criterion) among a list of options provided by the Wide-area Infrastructure Manager (WIM). It consists of three submodules that respectively handle: (*i*) the aggregation of vApp components based on affinities, (*ii*) the forecasting of (micro-) datacenter resources utilization, (*iii*) and the multi-site placement of the (aggregated) vApp micro-service components. The RSO's performance is mainly evaluated in terms of the execution times of its submodules while varying their respective input parameters, and additionally, three selection policies are also compared. Experimental results aim to highlight the RSO behavior in both execution times and deployment costs, as well as the RSO interactions with other OSS submodules and network platform components, not only for multi-site vApp deployment but also for other network/services management operations.**

***Index Terms*—5G, Multi-site resource allocation, Network slicing, OSS microservices, Resource selection, Vertical applications.**

I. INTRODUCTION

IN recent years, emerging network *softwarization* solutions such as the Multi-access Edge Computing (MEC), Network Functions Virtualization (NFV) and Software-defined Networking (SDN) paradigms, among others, have been established as key enablers of 5G technologies [1], along with the realization of a fully converged telecommunications infrastructure, able to simultaneously support a wide-range of *verticals* with highly heterogeneous nature and requirements.

The MEC paradigm [2] brings Cloud-like services closer to the end-users by deploying small- to medium-sized computing facilities, referred to as *micro-datacenters (μDC)* hereinafter, to support next-generation use-cases with challenging performance/operating requirements, such as the Ultra-reliable and Low-latency Communications (URLLC) use case category defined by the ITU-R [3]. Moreover, the μDCs in the edge and the remote Cloud DCs will play central and interworking roles in the softwarized network scenario as they will both potentially host (components of) the vertical applications (vApps), together with network services. With NFV [4], the latter are no longer limited to special-purpose physical network functions (PNFs) but also include Virtualized Network Functions (VNFs) – the software implementation of networking functionalities that run on general-purpose hardware and can be (dynamically) placed practically anywhere in the Cloud-MEC domain. While network/services management complexity is foreseen to escalate in this highly virtual environment, SDN [5] proves to be invaluable in terms of providing programmable interconnectivity – whether between vApp/service components, between infrastructure nodes and among them, which can also be exploited for implementing isolated network slices. As regards specifically vApps, they are 5G-ready applications that consist of several chainable cloud-native micro-services, i.e., components that have to collaborate in order to fulfil their operational scope, along with specific properties they should possess to be ported to the cloud. Collaboration implies that these components form a logical graph based on their dependencies. On top of that, the emergence of the programmable infrastructure has introduced additional characterizations that should be taken under consideration during a "strict" definition of a cloud-native application.

Along this line, *multi-tenancy* and the *as-a-Service* concept

Manuscript submitted on the 30th June 2021. This work has been partially supported by the Horizon 2020 5G-PPP Innovation Action 5G-INDUCE (Grant Agreement no. 101016941) and by the Horizon 2020 Innovation Action SPIDER (Grant Agreement no. 833685).

R. Bolla, R. Bruschi and F. Davoli are with the Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN) of the University of Genoa, and with the National Laboratory of Smart and Secure Networks (S2N) of the Italian National Consortium for Telecommunications (CNIT), Genoa, Italy (e-mail: roberto.bruschi@unige.it, franco.davoli@unige.it).

C. Lombardo is with the CNIT S2N National Laboratory, Genoa, Italy (e-mail: chiara.lombardo@cnit.it).

J. F. Pajo is with Telenor Research, Norway (e-mail: jane-frances.pajo@telenor.com).

are expected to evolve from the traditional Cloud virtualization to end-to-end network slicing, where 5G network slices [6] can involve both networking and computing resources from multiple domains and are exposed as slice abstractions of the underlying infrastructure. This will not only empower vertical industries, but also lower the barriers for start-ups and small/medium-sized enterprises (SMEs) to engage in the rollout and/or uptake of 5G technologies and services. In fact, as various business-to-business-to-X (B2B2X) opportunities will soon arise, service providers are foreseen to move up the value chain and, by 2030, they are forecasted to address around US$ 700 billion of global digitalization revenues across vertical industries [7].

This has motivated numerous research and/or innovation initiatives [8][9], fostering academic-industrial partnerships and cross-project synergies, towards building digital platforms that integrate verticals into the stakeholder ecosystem and could expose the converged infrastructure (based on Cloud, MEC, NFV and SDN technologies) to service providers, as well as in developing intelligent solutions for automating the deployment and orchestration of vApps and network services on a per-slice basis. In this respect, the H2020 5G-PPP MATILDA Project [10] sought to deliver an end-to-end 5G framework, which includes mechanisms for the instantiation of application-aware network slices, multi-site deployment of micro-service vApp components, as well as for the lifecycle management and orchestration of the slice resources, vApp components and P/VNFs involved. The MATILDA approach, , further enhanced in the subsequent project 5G-INDUCE [11],is based on the concept of separation of concerns between the Application Orchestrator in the vertical stakeholder's domain and the Network Orchestrator managed by the Telco Providers. A key role in the interaction of these two domains is played by the Telco Operations Support System (OSS).

This paper particularly looks into the *Resource Selection Optimizer (RSO)* of the MATILDA platform's OSS, and on its resource selection mechanisms among the Quality of Service (QoS)-aware, multi-site (and, possibly, multi-tenant) deployment options provided by a Wide-area Infrastructure Manager (WIM). Since the selection of the suitable resources is crucial for satisfying the requirements of vApps and particularly troublesome in a multi-tenant context, this paper aims to assess the suitability of the RSO to fulfil this task. Starting with a *slice intent* that encapsulates the vApp specifications (i.e., the vApp micro-service components, their interconnection, as well as the components' and their interconnecting infrastructural resources' QoS requirements, along with the necessary network services), the RSO is the software service in-charge of:

- aggregating vApp micro-service components according to their affinities, and to a set of QoS constraint thresholds, defined by the Application Service Provider, on their interconnecting infrastructure (which can be abstracted in terms of logical "links" in the micro-service chain that makes up the application service);
- forecasting the resources' utilization (such as the amount of free resources of vCPU, RAM and disk, as well as their usage) in the μDCs; and
- placing the (aggregated) vApp micro-service components according to their requirements, the deployment options provided by the WIM and the utilization forecasts in the μDCs involved.

Tests performed on the three submodules composing the RSO allow identifying the different behaviors and trade-offs that can improve the interplay with the network platform components as well as selecting the setup that is the most appropriate to the current network environment and application requirements.

The remainder of this paper is organized as follows. Section II provides a summary of the most related works in the literature, while Section III describes the architecture, deployment and lifecycle management of vApps in the context of MATILDA. The RSO's design and operation are then detailed in Section IV, followed by the performance evaluation results and discussion in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work

Resource allocation problems in the literature have been recently evolving with an end-to-end notion, as 5G-ready applications and their underlying network slices are expected to span multiple domains.

The authors in [12] proposed a framework for the management and deployment of the 5G core network, which considers the distributed deployment of the functions among μDCs; the optimal allocation is a mixed nonlinear integer programming problem and is driven by the μDCs' energy costs and processing delays, as well as the delay and bandwidth costs in the backhaul. On the other hand, an end-to-end slicing framework is proposed in [13], considering both computing and communication resources across the full 2-tier MEC architecture; each service/slice is allocated resources that are merely sufficient to meet its latency requirements.

In [14], the authors proposed a heuristic for selecting the network resources for the slice, based on the utility scores of the candidate resources that take into account end-to-end availability, reliability and delay constraints. Complex network theory is adopted in [15] to obtain the topological information of slices and infrastructure network, which are then used to define a node importance metric for mapping network slice requests to the infrastructure. In [16], a resource allocation model for 5G network slices is proposed as a convex optimization problem that maximizes the overall system utility, taking into account a resource demand vector for the VNFs involved in the slice; a distributed solution is further achieved through an auction-based approach that maps a system of collaborative slices among DCs. Moreover, an optimal model for cross-domain network slices deployment is presented in [17], which jointly considers the constraints on layered placement, resources, link arrangements, latency and bandwidth; the authors also proposed a heuristic based on a distributed multi-layer knapsack problem, in which items (e.g.,

VNFs or VNF components) cannot be packed unless the required communication resources are met.

Numerous approaches (e.g., [18]-[23], among others) based on network embedding have also been published in recent years. In particular, the authors in [18] proposed a virtual network embedding problem based on 3-D resources that includes computing, network, and storage; [19] looked into a network slice embedding problem that considers the deployment costs due to the slice's minimum resource requirements (both at the nodes and links), as well as a cost related to the end-to-end delay (i.e., propagation and processing delays plus the virtualization overhead); [20] proposed an efficient heuristic for network slice embedding that allocates the network slice resources based on node rankings (four possible ranking algorithms are evaluated); and [21] proposed an algorithm for automatic virtual network embedding based on deep reinforcement learning with a novel multi-objective reward function. Meanwhile, [22] and [23] investigated the partitioning of multi-domain virtual networks and proposed heuristics based on particle swarm optimization; the latter also proposed a two-step embedding strategy for the inter- and intra-domain embedding sub-problems.

Other works that consider affinity among service components include [24] and [25]. The former focused on jointly minimizing the deployment cost for service providers, response times and inter-cloud traffic in a multi-cloud scenario through an integer linear programming problem; a heuristic based on VNFs' traffic affinity is also proposed for service chain placement. In the context of user-centric virtual networks, the latter proposed a proximity- and affinity-aware clustering of virtual objects for scalable management.

This paper proposes the Resource Selection Optimizer (RSO), a software-service in the MATILDA OSS [26] [27] that performs the selection of resources inside the virtual infrastructure and across the wide-area interconnected infrastructure. The three submodules composing the RSO, and the heuristics they enforce, aggregate the micro-service components of a vApp graph based on affinity (i.e., the delay, jitter, packet loss and throughput requirements between them), obtain a list of QoS-aware deployment options from the WIM, and selects the placement of the resulting reduced graph from this list based on the deployment costs, the number of μDCs involved and the utilization forecasts in the μDCs.

The characteristics of the proposed service that mainly differentiate it from the other approaches that we have outlined regard the multi-tenancy and the possibility of creating interconnected deployments spanning multiple tenant domains, as well as how the forecasting and deployment decisions explicitly account for the need to deploy Network Services (NSs) alongside the vApp components in the μDCs/DCs composing the infrastructure. Details on how this goal is achieved, as well as the role of the RSO in the creation of a network slice, will be presented in Section IV.

III. 5G-Ready vApps: Architecture, Deployment and Lifecycle Management

This section provides a brief background on the architecture of 5G-ready vApps, as well as how their multi-site deployment and lifecycle management can be supported over virtualized infrastructures.

A. Microservices Architecture

The microservice architectural style has been gaining momentum in the research and industry scenes for benefits such as improved agility, scalability and autonomy of softwarized services' components [28]. This approach is currently being adopted in the design and development of 5G-ready applications and network services, whose component instances are foreseen to be deployed and scaled among multiple geo-distributed μDCs in the edge (and remote DCs) according to demand dynamics.

On the applications' perspective, microservices have been around in the last decade, with architectures evolving with the software technologies – from hard-coded container-based to serverless applications [28]. With the emergence of SDN and NFV, this concept has been also extended to network services. In fact, the ETSI 3GPP has defined the 5G system with a service-based architecture (SBA) [29], such that 5G network services, as well as the underlying control and user plane functions (C/UPFs), are designed as chains/meshes of microservices. Moreover, the smooth rollout of 5G involves interworking with 4G technologies in the next years, which means that 5G-ready applications and their (wide-area) interconnectivity will be jointly supported by 4/5G P/VNFs.

B. Network Slicing

The NGMN and 3GPP also introduced the network slicing concept into 5G ecosystem specifications [30] [31], where a network slice can be roughly summarized as a virtual projection of a 5G network with all the functionalities, isolation level, and capabilities customized according to the needs of the vertical applications. This would further facilitate evolution towards a multi-service ecosystem that would simultaneously support highly heterogeneous application-specific requirements in a differentiated manner.

In this scenario, the telecommunications infrastructure emerges as a sort of distributed datacenter with an aggregate pool of networking / computing / storage / radio resources – the challenge remains in the coordination and autonomous control of this pool in realizing and managing end-to-end network slices. It is worth noting that the latter encompass both the networking and computing domains, in which applications and network services are orchestrated via the slice abstractions. This way, vertical industries and service providers can respectively manage the complete lifecycle of their application graphs and network service chains without needing to know the underlying infrastructure – starting from planning (*Day-0*), initial deployment (*Day-1*), through the in-life operations (*Day-2*).

C. The MATILDA Platform

The MATILDA Project aimed at providing an end-to-end 5G framework for the *design*, *development*, *deployment* and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3151468, IEEE Transactions on Network and Service Management

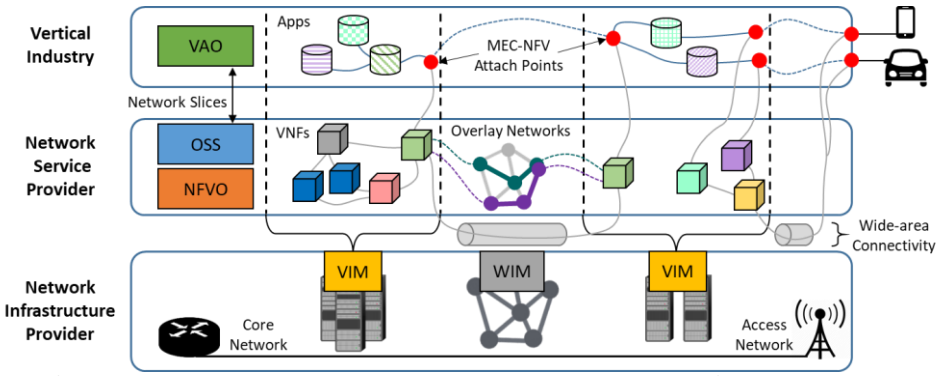


Fig. 1. Deployment of 5G-ready applications spanning multiple VIMs over the evolved network infrastructure with the MATILDA platform.

orchestration of microservice-based 5G-ready applications over a programmable infrastructure. This also involves the *instantiation* and *management* of application-aware network slices, including network and computing resources, as well as the necessary network services therein (i.e., P/VNFs).

As illustrated in Fig. 1, there are three main stakeholders involved in MATILDA's layered approach: (*i*) the **Vertical Industry**, (*ii*) the **Network Service Provider**, and (*iii*) the **Network Infrastructure Provider**. The vertical industry drives the business value of the 5G-ready applications, which is taken into account in the design and development, as well as in the compute/networking requirement specifications. The network service provider then enables the necessary 4/5G services (i.e., chaining of P/VNFs) to support the applications, and the interconnectivity among their components. Finally, the network infrastructure provider offers the underlying network and computing resources for the materialization of the application-aware network slices.

The main architectural blocks of the MATILDA platform include:

- the **Vertical Application Orchestrator (VAO)**, in charge of the lifecycle management of the application graphs, as driven by vertical industry requests and infrastructure-level events;
- the **NFV Orchestrator (NFVO)**, in charge of the lifecycle management of application-/slice-specific network services;
- the **Wide-area Infrastructure Manager (WIM)**, which maintains a complete knowledge of the status of the wide-area transport network and the geo-distributed infrastructure resources, which are essential for creating the network slices' overlay networks;
- the **Virtual Infrastructure Managers (VIMs)**, which abstract and expose computing, storage, and networking capabilities of each µDC or DC within the evolved network infrastructure; and
- the **Operations Support System (OSS)**, in-charge of coordinating the operations of the aforementioned blocks in order to properly set up the application-aware network slices and support the (geo-distributed) applications.

Note that all platform components and their reference points are fully compliant with ETSI NFV architectural specifications [32]. Moreover, with the ETSI MEC WG pointing out how the user planes of the application and network services could be hosted in different isolated tenant spaces within the VIMs [33], [34], MEC-NFV *attach points* (e.g., the virtual networks interconnecting application components and VNFs hosted in the VIM) are realized as virtual networks interconnecting the application and NFV domains. Fig. 1 depicts a deployment example of 5G-ready applications that span multiple VIMs, also putting in evidence where the different platform components reside. It is worth noting that, in MATILDA, the WIM was based on the Ericsson Network Manager [35] whose internal structure cannot be disclosed. However, a high level description of its interaction for the materialization of a slice is reported in the next section.

IV. The Resource Selector Optimizer

The MATILDA OSS is designed as a mesh of micro-services, in which all software services are independently deployable, parallelizable and maintain their states in an external database. With this in mind, a working prototype has been developed as a suite of four software services and two databases, as illustrated in Fig. 2.

In a nutshell, the software suite includes: (*i*) the *Slicing Northbound* module that supports the communication between the VAO and OSS to enable the deployment and lifecycle management of application components; (*ii*) the *Resource Selection Optimizer (RSO)* module that provides the resource allocation and reinforcement algorithms/policies regarding the multi-site deployment of the applications' components; (*iii*) the *NFV Convergence Layer (NFVCL)* module that provides a level of abstraction to enable the complete lifecycle management and orchestration of network services and P/VNFs instantiated in the 5G infrastructure; and (*iv*) the *OSS Core* module that contains the VIM and WIM convergence layers to provide the

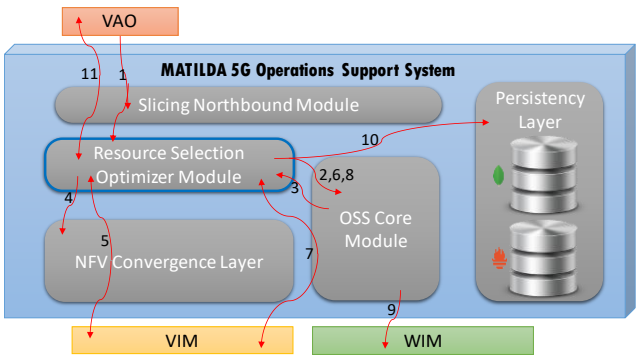


Fig. 2. The MATILDA OSS Architecture.

external interfacing of the OSS to the VIMs and WIM(s). Moreover, the *Persistency Layer* includes two different databases based on MongoDB [36] and Prometheus [37] for storing the configuration and monitoring data, respectively.

Upon the reception of a slice intent from the VAO, the above mentioned micro-services interact to produce a candidate materialized slice according to the following procedure:

1. The Slicing Northbound module receives a slice intent message from the VAO. The message is parsed and checked, and the slice intent data is stored in the Persistency Layer; then, a trigger message is sent to the RSO including the reference to the stored data.
2. The RSO analyses the application graph included in the slice intent, and applies a graph reduction algorithm to group the components into "macro nodes" that need to be placed in the same VIM according to the thresholds provided by the VAO. Then, it requests a list of candidate VIMs for the deployment of all the groups to the WIM Convergence Layer in the OSS Core. It also keeps track of the monitored resources' utilization and provides forecasts.
3. The WIM Convergence Layer calculates a number of possible deployment options and sends them back to the RSO.
4. The RSO uses the slice intent data and the reduced graph to select the network services to be deployed to materialize the slice. The RSO asks the NFVCL for the quota of resources to be reserved for the NSs in each VIM.
5. The NFVCL handles the request from the RSO by mapping the type of NS against the ones available in the selected VIMs, and extracts the amount and type of resources to be reserved for each VIM.
6. The RSO aggregates the amount and the type of resources to be reserved for both the vertical application components and the VNFs, and it asks the VIM Convergence Layer in the OSS Core for the availability of such resources at the selected VIMs.
7. The VIM Convergence Layer checks for the availability of the resources at the selected VIMs. The VIM Convergence Layer replies to the RSO with an availability report, and if all the requirements are satisfied, it reserves the required resource quotas.
8. The RSO handles the reply from the VIM Convergence Layer. If one or more VIMs cannot provide the required types or amount of resources, then the RSO will restart from Step 4 with a different deployment option. If all the requests are satisfied, the RSO asks the WIM Convergence Layer to reserve the wide-area resources for the chosen deployment option.
9. The WIM Convergence Layer reserves the wide-area resources (i.e., paths) for the chosen deployment option.
10. The RSO stores in the Persistency Layer all the required data to fill the candidate materialized slice message, and to proceed to the slice set up in case of positive response from the VAO.
11. The Slicing Northbound Module collects the data stored by the RSO, parses them into a candidate materialized slice message, and sends it to the VAO.

With respect to the above procedure, this work particularly looks into the design and operational details of the RSO, which consists of three submodules that respectively handle the vApp *graph reduction*, VIM resources *utilization forecasting* and multi-site *placement optimization* of the reduced graph. Since the algorithms/policies adopted may have heavy computational requirements, can run in parallel, and should provide results with short and finite time horizons, the RSO is designed to scale horizontally. Multiple instances of the RSO can register themselves against the OSS Core module, by indicating which algorithms and/or policies they provide. When a certain algorithm/policy has to be applied in a workflow, the OSS Core triggers the proper RSO instance providing it.

Each RSO instance is based on an image of the open-source SageMath project [38]. By adopting such an advanced mathematical framework (which integrates software packages in R and Python), it is possible to avoid the porting of the algorithms from an environment where they are designed and initially tested through simulations (like MatLab, Mathematica, etc.), and to directly use them in the production environment. This way, the integration effort (and related debugging) can be almost zeroed, while providing a familiar and comfortable ecosystem to mathematicians and scientists working on algorithm design.

Furthermore, the base image containing SageMath has been extended with agents and libraries to communicate with the OSS Core and the Persistency Layer. As far as the latter is concerned, the RSO can access: (*i*) the MongoDB database to write generated data, or to retrieve requirements and configuration data on the slice intents, on the services, and on the infrastructure elements; (*ii*) the Prometheus database to retrieve performance metrics stored by the OSS monitoring framework.

The following subsections describe the three RSO submodules. Table I reports the notation that will be adopted in the remaining of Section IV.

A vApp has N nodes (corresponding to micro-service components) and L links (specifying the nodes' logical interconnection), the graph specifications define the nodes $\{node_n\}$, $n = 1, \dots, N$, links $\{link_l\}$, $l = 1, \dots, L$, and the link

TABLE I. NOTATIONS DEFINITION

$node_n$	One of the nodes composing the vApp graph, $n = 1, \dots N$
$link_l$	One of the links composing the vApp graph, $l = 1, \dots L$
$\{e_l^{node_n}, e_l^{node_o}\}$	The couple of nodes representing the endpoints of $link_l$
$cmetric_c$	One of the $c = 1, \dots, C$ performance metrics for the vApp
$cval_l$	Constraint value for $cmetric_c$, the set being $cmetric_c$: $[cval_1, \dots, cval_L]$
$mnode_m$	Macro nodes composing the reduced graph, $m = 1, \dots M$
$rval_m$	Set of requirements for vApp and NS for each resource type $'vcpu', 'ram', 'disk'$
$dopts_d$	Deployment option, , $d = 1, \dots D$
V_m^d	VIM that will host macro node $mnode_m$ in the deployment option $dopts_d$
$Acost_v$	Agreement cost for deployment option v
α_v	Ratio between the usage and the overcommit ratio
$Dcost_d$	The cost of a deployment option $dopts_d$
η_{d}	Number of VIMs involved in a deployment option

endpoints e_l are couples of nodes edges $\{e_l^{node_n}, e_l^{node_o}\}, n, o \in N$. Constraint values $\{cval_l\}$ are indicated according to different subsets of constraints on performance metrics $\{cmetric_c\}$ such as delay, jitter, packet loss and/or throughput, $c = 1, \dots, C$. The vApp graph is then given as

$$Graph = [[node_1, \dots, node_N], [link_1, \dots, link_L], \left[\{e_1^{node_n}, e_1^{node_o}\}, \dots, \{e_L^{node_n}, e_L^{node_o}\}\right], \{cmetric_c: [cval_1, \dots, cval_L], \dots\}] \quad (1)$$

On the other hand, the constraint thresholds $\{cthresh_c\}$ (i.e., upper bounds on the corresponding vApp graph links' QoS constraints) can assume adaptable values, as defined by the application service provider.

The reduced graph obtained in Step 2 has M macro nodes defined by $\{mnode_m\}$, $m = 1, \dots, M$, with corresponding application and NFV domain requirements $\{rval_m\}$ for each resource type $\{'vcpu', 'ram', 'disk'\}$; then the slice requirements are given as

$$Slice\_reqs = [[mnode_1, \dots, mnode_M], \{'vcpu': [rval_1, \dots, rval_M], 'ram': [rval_1, \dots, rval_M], 'disk': [rval_1, \dots, rval_M]\}] \quad (2)$$

The deployment options calculated in Step 3 by the WIM Convergence Layer are given as tuples of VIMs mapped with the vApp graph (macro) nodes. For instance, if D deployment options are given for M macro nodes, we have:

$$Deployment\_opts = [(V_1^1, \dots, V_M^1), \dots, (V_1^D, \dots, V_M^D)] \quad (3)$$

where V_m^d, $m = 1, \dots, M$, $d = 1, \dots, D$, is the VIM that will host macro node $mnode_m$ in the deployment option $dopts_d$. Note that $V_{m1}^d \equiv V_{m2}^d$ if $mnode_{m1}$ and $mnode_{m2}$, $m1, m2 \in \{1, \dots, M\}$, are hosted on the same VIM in deployment option $dopts_d$.

A. The Graph Reduction Submodule

Given the graph specifications in the slice intent and a set of graph link QoS constraint thresholds, this submodule aggregates vApp components according to the latter, generating a reduced graph of *macro nodes* such that all components in the same macro node will be collectively handled in the subsequent deployment actions on µDCs/DCs in the infrastructure. The goal is to ensure that the same macro-nodes will be treated as an inseparable set in the following placement and deployment operations/actions. Since the threshold parameters determine which components are part of the same macro-node and, then, need to be placed in the same VIM, tuning these parameters can result in different outcomes. Therefore, these values need to be carefully pondered, for instance by considering the peculiarities of the telecom infrastructure (e.g., the average end-to-end delays among the VIMs). It is worth noting that computing parameters are taken into account in the next stages.

The RSO adopts a fairly straightforward graph reduction

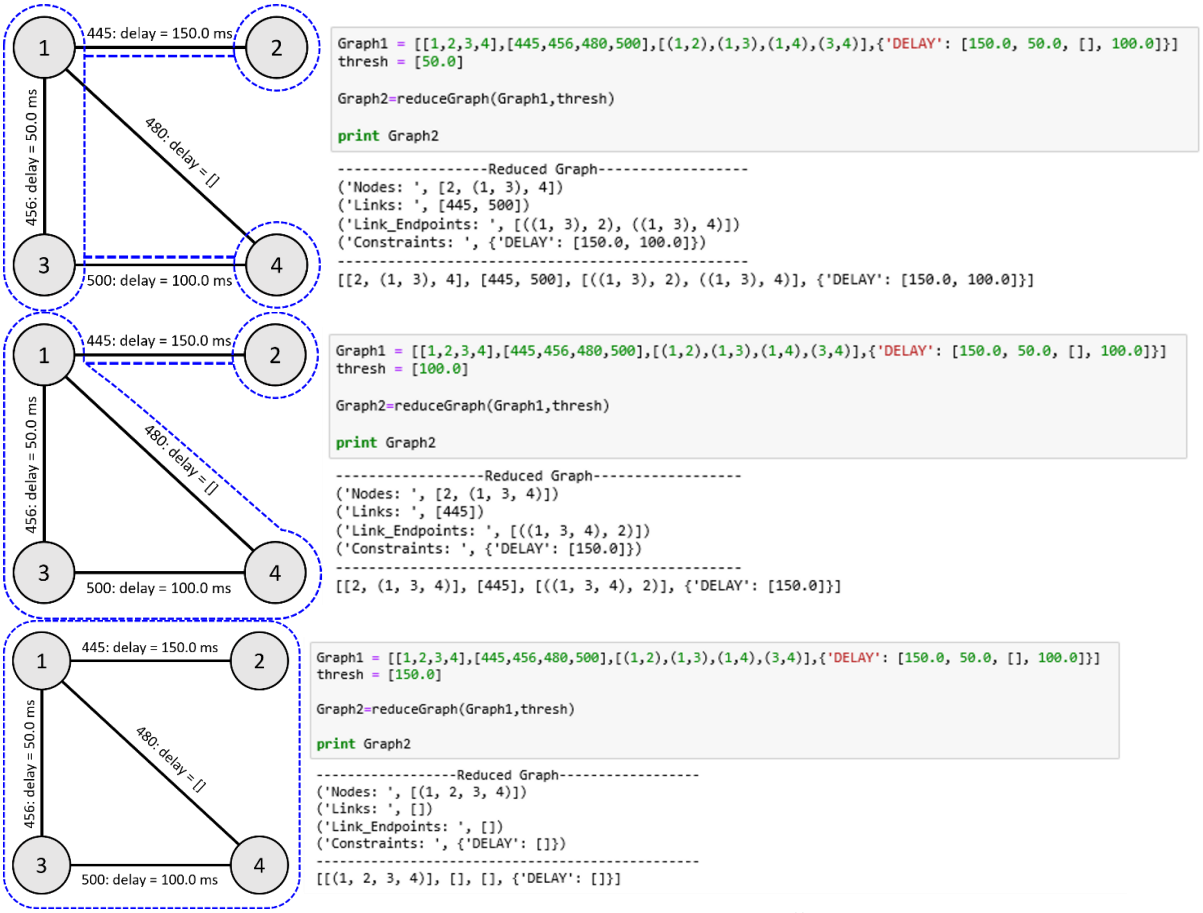


Fig. 3. vApp graph reduction results according to different threshold values.

policy that jointly considers the vApp graph specifications provided by the VAO, and groups together the nodes with logical link constraint values that meet the thresholds (i.e., $delay \leq Delay\_th$, $jitter \leq Jitter\_th$, $packet\ loss \leq PacketLoss\_th$ and $throughput \geq Throughput\_th$) as macro nodes. The output is a reduced graph, whose specification is defined in a similar format as the input graph specifications; it will then be passed to the WIM convergence layer that will provide a list of deployment options, each one specifying the potential VIMs that could host the (macro) nodes of the vApp graph.

Considering the case of a slice intent with $cmetric_c = delay$ as the only constraint, we can write the pseudocode reported in Algorithm 1:

Algorithm 1 GraphReduction

n nodes in $\mathcal{N} = \{1, \ldots, N\}$, l links in $\mathcal{L} = \{1, \ldots, L\}$
$\{e_l^{node_n}, e_l^{node_o}\}$ endpoints
macronode $mnode_m$
for l in $\mathcal{L}$ **do**
  if $delay_1 \leq Delay_{th}$ **then**
    Append $(node_n^1, node_o^1)_l$ to $mnode_m$
    Remove l from L
  end if
end for

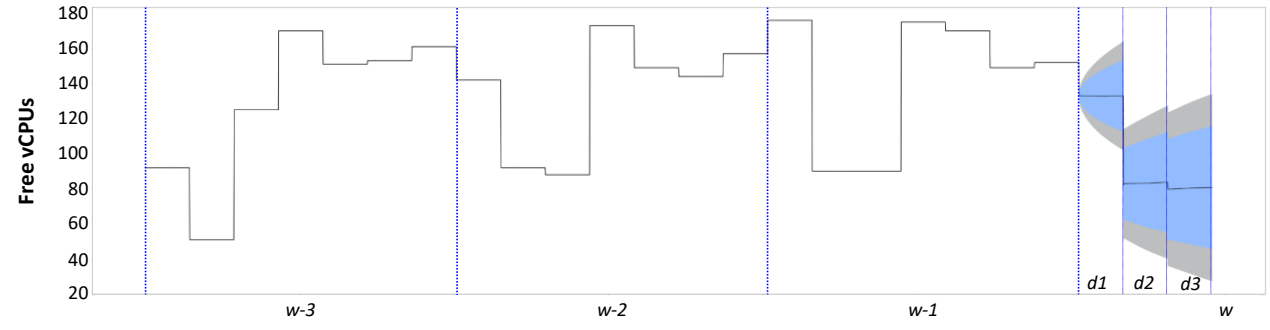
For instance, consider a simple slice intent with 4 application components (e.g., nodes) {1, 2, 3, 4} interconnected by 4 links labelled as {445, 456, 480, 500}, with only delay constraints (in ms). Fig. 3 shows how the resulting reduced graph changes with the threshold. The graph reduction affects both nodes and links, as some may be no longer required (for instance, see link (1,4) in the first example) once two nodes are merged.

Undefined parameters are treated as "wildcards" and as such their assignment is determined by the peculiarities of the telecom infrastructure. Moreover, in the case that more constraint metrics are specified in the slice intent, the link constraints will be updated with the union of the most stringent requirements for each metric.

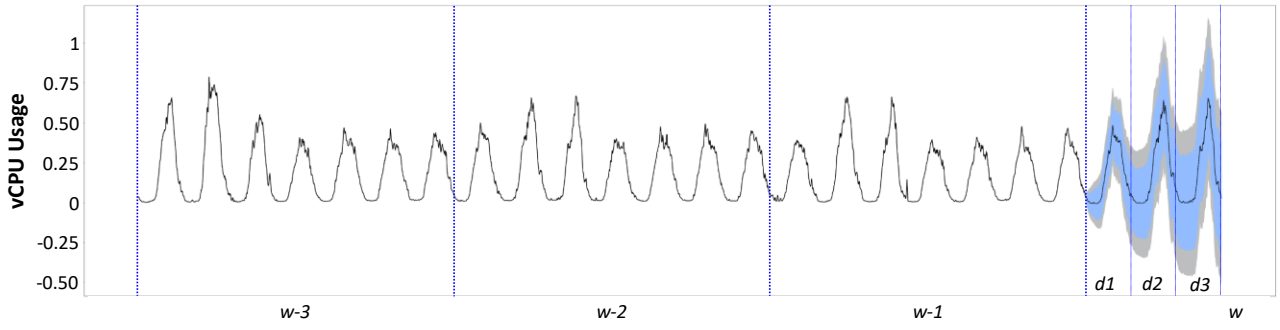
B. The Utilization Forecasting Submodule

Given the list of candidate VIMs provided by the WIM convergence layer, it is necessary to first define a set of metrics to be used in the selection of the most suitable VIMs for the (macro) nodes. With this in mind, we consider the monitoring metrics on vCPU, RAM and disk utilization of the candidate VIMs collected on the MATILDA platform available in the Prometheus database for the previous observation period (e.g., time series data of the last three weeks), such as the amounts of free resources and actual usage.

This submodule has two functions: (*a*) the `modeling`, and (*b*) the `forecasting` functions. The former runs periodically in the background for keeping the resource models up-to-date, while the latter can be called at runtime (or also periodically in the background, since we consider either the *Maximum* values or the *Quantiles* of the forecasts rather than the current values of the monitoring metrics, in order to provide the necessary headroom in the allocation of resources, while supporting different dynamics in the time series) to generate utilization forecasts for VIM resources based on the most recently updated models. Either way, the execution times of this submodule do not impact on the overall slice creation described in Section IV. Moreover, *thread-based parallelism* is adopted to update the



(a). # of free vCPUs historical time series and forecast



(b). vCPU usage historical time series and forecast

Fig. 4. Multi-seasonal vCPU utilization forecasting example with observation period set to three weeks and forecasting horizon to three days.

models with a running window of training data in the background, as well as to use the most recently updated models for the forecasts.

With R's `forecast` library, this submodule models the *multi-seasonality* (i.e., daily and weekly) of the time series data using the `msts` function. The resulting models are in turn used as input to the `forecast` function to predict the time series values for a certain horizon (e.g., for the next three days).

Five different forecasting methods have been tested on three different time windows, e. g., one, three and seven days., with samples taken every ten minutes. While the execution time of the individual methods does not show significant differences, and the main difference, as will be shown in Section V, depends on the level of parallelism, we used a "Multiple Seasonal Holt-Winters" method [39], which represents an extension of Holt-Winters that not only captures Weekday/Weekend differences, but also supports multiple seasonalities (e.g., day, week, month,...).

Since, as will be shown in the results, the forecasting function runs for over an order of magnitude longer than the modeling function, running the two sequentially allows for forecasts that are always up-to-date.

For instance, Fig. 4 shows the resulting forecasts regarding the vCPUs of a candidate VIM in terms of the number of free vCPUs and VIM-wide usage. In all the tests performed on the available traces, 99% of the measured values fall within the confidence interval, with an error below 5% of the forecasted values.

C. The Placement Submodule

Finally, given the aggregate computing-network slice requirements (i.e., macro nodes plus network services provided by the NFVCL in Step 5), the list of deployment options (i.e., the mapping between the macro nodes and VIMs) from the WIM convergence layer in Step 3 and the corresponding utilization forecasts (in terms of Maximum values or the Quantiles) of the candidate VIMs, this submodule seeks to select the most suitable deployment of the slice.

In more detail, the slice requirements are derived from the minimum amount of resources (i.e., vCPU, RAM and disk) required by the (components of the macro) nodes of the vApp graph and by the necessary network services (depending on the network services available in the candidate VIMs), as specified in the slice intent.

Given the set of VIMs involved in the deployment options, the information on the necessary VIM resources includes the Maximum values (or the Quantiles) of the forecasted amount of free resources and usages, as well as their pre-defined overcommit ratios (e.g., the OpenStack compute service uses the default overcommit ratios of 16, 1.5 and 1 for CPU, RAM and disk, respectively [40]), and *agreement* costs (e.g., the cost applied by the owner of the VIM to give access to its resources); the latter are particularly useful for supporting scenarios in which the candidate VIMs have different owners and/or costs. Suppose that there are Υ candidate VIMs in the deployment options defined by $\{V_v\}$, each corresponding to an agreement cost $Acost_v,\ v = 1, \dots, \Upsilon$; the VIM resources information is collectively expressed as:

$$\begin{aligned} VIMs\_res \ = \ [&[V_1, \dots, V_\Upsilon], [Acost_1, \dots, Acost_\Upsilon], \\ &\{'vcpu\_free': [rval_1, \dots, rval_\Upsilon], \\ &\ 'ram\_free': [rval_1, \dots, rval_\Upsilon], \\ &\ 'disk\_free': [rval_1, \dots, rval_\Upsilon]\}, \\ &\{'vcpu\_usage': [rval_1, \dots, rval_\Upsilon], \\ &\ 'ram\_usage': [rval_1, \dots, rval_\Upsilon], \\ &\ 'disk\_usage': [rval_1, \dots, rval_\Upsilon]\}, \\ &\{'vcpu\_ocratio': [rval_1, \dots, rval_\Upsilon], \\ &\ 'ram\_ocratio': [rval_1, \dots, rval_\Upsilon], \\ &\ 'disk\_ocratio': [rval_1, \dots, rval_\Upsilon]\}] \end{aligned} \quad (4)$$

With the goal of drawing the line between deployment options, the coefficient α_v is further defined as the ratio between the usage and the overcommit ratio computed for the three resource types $x \in \{'vcpu', 'ram', 'disk'\}$ in a VIM as

$$\alpha_v = \left(\alpha_v^{vcpu}, \alpha_v^{ram}, \alpha_v^{disk}\right) \quad (5)$$

where $\alpha_v^x = x\_usage/x\_ocratio$. These coefficients will act as weighting factors (dynamically) differentiating the costs of VIM resources, such that their low usage and/or high overcommit ratio suggest lower costs.

1) Decision Rules

In this work, two decision rules are considered in the development of algorithms/policies for the selection among vApp graph deployment options.

R1. Minimizing the cost of the required resources

The cost $Dcost_d$ of a deployment option $dopts_d$ is derived firstly with a two-step aggregation approach, where the macro node requirements per resource type are summed up according to the hosting VIM V_v, multiplied by their respective α_v^x *and* $Acost_v$*;* then, the weighted requirements per resource type are again summed up for each deployment option. Subsequently, the resulting values for all D deployment options are normalized, still per resource type, before computing their corresponding square resultants. $Dcost_d$ is then given by

$$Dcost_d = \sum_x \left(\left\| \sum_{V_v \in dopts_d} Acost_v \ \alpha_v^x \sum_{mnode_m \mapsto V_v} rval_{mnode_m} \right\|_D \right)^2 \quad (6)$$

where $\|\cdot\|_D$ is the normalization operator with respect to all deployment options, and $mnode_m \mapsto V_v$ identifies all macro nodes $mnode_m$ that are mapped to the VIM V_v.

R2. Minimizing the number of VIMs involved

The number of VIMs involved in a deployment option, η_d, can range from 1 through M – the former means that all macro nodes are hosted on the same VIM, while the latter means that each one of them is hosted on a different VIM. By minimizing the number of VIMs involved, we seek to also minimize the complexity of managing the interconnectivity among the macro nodes. Two approaches can be followed to incorporate this rule; the number of VIMs involved per deployment option can be used: (*i*) as a weighting factor to Eq. (6) prior to the minimization of rule **R1**, and (*ii*) as the cost of an independent minimization rule that can be performed before/after rule **R1**.

2) Selection Policies

Three policies are then derived to jointly consider both ***R1*** and ***R2*** in a multi-objective optimization whose solution is performed in a heuristic manner as a combinatorial problem*; the policies are differentiated according to how the rules are incorporated in the implementation.*

P1. MinReqMinV

This policy selects the deployment option with the minimum of the deployment cost multiplied by the number of VIMs involved per deployment option:

$$Dcost_d{}' = \eta_d Dcost_d \quad (7)$$

If multiple solutions exist, one is randomly chosen according to a uniform distribution. Algorithm 2 illustrates the policy:

Algorithm 2 MinReqMinV

η_d in $\{\eta_1, \dots \eta_D\}$
$Dcost_d$ in $\{Dcost_1, \dots Dcost_D\}$
V'_d selected options
for d in D **do**
  if $\eta_{d+1} Dcost_{d+1} < \eta_d Dcost_d$ **then**
    $\{V'_d\} = V_{d+1}$
  end if
end for

P2. MinReq+MinV

This policy sequentially executes the minimization of $Dcost_d$, and then of η_d. If a unique solution is found in the first minimization step, there is no need to execute the *Else* statement and perform the second *Sort*. If multiple solutions exist after the two steps, one is randomly chosen according to a uniform distribution, as shown in Algorithm 3.

Algorithm 3 MinReq+MinV

η_d in $\{\eta_1, \dots \eta_D\}$
$Dcost_d$ in $\{Dcost_1, \dots Dcost_D\}$
V'_d selected options
$Dcost_d^{sorted}$ = **sort** $(Dcost_d)$
if $Dcost_d^{sorted}[1] < Dcost_d^{sorted}[2]$ **then**
  $V'_d = V_d^{sorted}[1]$
else
  for each $Dcost_d^{sorted}[i] == Dcost_d^{sorted}[1]$ **do**
    sort η_d^i
  end for
end if

P3. MinV+MinReq

This policy is similar to ***P2***, but with a reversed sequence. Particularly, the minimization of η_d is first executed, to find the option with the minimum number of VIMs involved, and then that of $Dcost_d$ (to find the solution with the minimum $Dcost$ in the subset, if the first solution is not unique). If a unique solution is found in the first minimization step, there is no need to execute the second one. If multiple solutions exist after the two steps, one is randomly chosen according to a uniform distribution (see Algorithm 4).

As previously anticipated, various RSO instances may adopt different algorithms/policies, which are not limited to the ones evaluated in this study, nor to the decision rules considered.

Algorithm 4 MinV+MinReq

η_d in $\{\eta_1, \dots \eta_D\}$
$Dcost_d$ in $\{Dcost_1, \dots Dcost_D\}$
V'_d selected options
η_d^{sorted} = **sort** (η_d)
if $\eta_d^{sorted}[1] < \eta_d^{sorted}[2]$ **then**
  $V'_d = V_d^{sorted}[1]$
else
  for each $\eta_d^{sorted}[i] == \eta_d^{sorted}[1]$ **do**
    sort $Dcost_d^i$
  end for
end if

V. Performance Evaluation

The performance of the RSO is mainly evaluated in terms of the execution times of its submodules – *Graph Reduction*, *Utilization Forecasting* and *Placement Optimization* – while varying their respective input parameters. For the latter, the deployment costs corresponding to the three selection policies are also compared. Since the submodules are executed sequentially, the analysis is done in a similar fashion. A breakdown of the impact of the RSO within the procedure described in Section IV can be found in [26].

A. Graph Reduction

Random vApp graph topologies are considered in this evaluation – particularly, graphs with $N = \{2, 5, 10, 15, 20\}$ nodes and links L ranging from $L_{min} = N - 1$ through $L_{max} = N(N-1)/2$ have been generated. As previously anticipated, different subsets of constraint metrics $\{cmetric_c\}$ (e.g., delay, jitter, packet loss and throughput) can be specified in the slice intent, hence defining different combinations with 1 to 4 constraint metrics. All constraint values are drawn from discrete uniform distributions. *Delay* constraint values from $D_{dist} = U\{25, 50, 75, 100, 125, 150\}$ ms; *jitter* constraint values from $J_{dist} = U\{5, 10, 15, 25, 30\}$ ms; *packet loss* constraint values from $P_{dist} = U\{0.0025, 0.005, 0.0075, 0.01, 0.0125, 0.015\}$, and; *throughput* constraint values from $T_{dist} = U\{0.5, 1, 1.5, 2, 2.5\}$ Gbps. The tests are repeated for 10 runs with varying seeds, generating a different topology and/or constraint values for each (N, L) combination.

Based on the constraint values' distributions above, four different threshold levels are defined and considered in the evaluations.

***T1.* In=Out**

$Delay\_th < D_{dist}$, $Jitter\_th < J_{dist}$, $PacketLoss\_th < P_{dist}$ and $Throughput\_th > T_{dist}$, such that no logical link constraint values meet the thresholds – hence the input and output vApp graphs are the same.

***T2.* Min**

$Delay\_th = \min(D_{dist})$, $Jitter\_th = \min(J_{dist})$, $PacketLoss\_th = \min(P_{dist})$ and $Throughput\_th = \max(T_{dist})$, such that the number of logical link constraint values that meet the thresholds per metric is minimized.

***T3.* Med**

$Delay\_th = \text{median}(D_{dist})$, $Jitter\_th = \text{median}(J_{dist})$, $PacketLoss\_th = \text{median}(P_{dist})$ and $Throughput\_th = \text{median}(T_{dist})$, such that logical link constraint values have more/less 50% probability to meet the thresholds.

***T4.* Max**

$Delay\_th = max(D_{dist})$, $Jitter\_th = max(J_{dist})$, $PacketLoss\_th = max(P_{dist})$ and $Throughput\_th = min(T_{dist})$, such that all the logical link constraint values meet the thresholds.

Fig. 5 shows how the average execution times of the **Graph Reduction** submodule vary with N for each threshold level. Two different trends can be observed that correspond to the cases $N = 2$ and $N > 2$. The first case is simply a linear trend – which is expected, since there is only 1 logical link and the probability that its constraint value(s) meet the threshold(s) increases with the threshold level – and results in a binary decision (i.e., all or none) on executing the code for merging the nodes. On the other hand, the second case is trickier, since merging nodes can result in a chain effect that highly depends on the vApp graph topology. Starting from the **In=Out** threshold level that does not merge any nodes, the **Min** level result in more involved chained effects that, on average, increase with N (considering all possible values of $L \in [L_{min}, L_{max}]$); more involved in the sense that the size (M and L') of the reduced vApp graph is not greatly decreased, and the logical link specifications need to be updated. Then, as the threshold level is further increased to **Med** and to **Max**, M and L' decrease accordingly and, hence, the execution time of the code for updating logical link specifications is also reduced. Note that each point of the curves Fig. 5 is averaged not only over all possible values of $L \in [L_{min}, L_{max}]$, but also over different numbers of constraint metrics.

B. Utilization Forecasting

The execution times of both the modeling and forecasting functions of the Utilization Forecasting submodule are evaluated individually and jointly at the resource and VIM levels, for varying number of VIMs involved in the deployment options. As shown in Fig. 6, the forecasting function runs for over an order of magnitude longer than the modeling function – on average, building/updating a resource model takes around 0.09 s, while generating a forecast takes around 4.42 s. This provides an indication for deciding how often they should be run. Since updating the model does not take so long, it may be reasonable to also run them sequentially to generate updated forecasts. Moreover, with thread-based parallelism, the values at the VIM level can correspond to those at the resource level if the modeling and/or forecasting for VIMs' vCPU, RAM and disk resources are run in parallel; otherwise, the average time required for their sequential execution linearly increases with the number of resource types per VIM (i.e., approximately 3×, given the three resource types considered).

Fig. 7 illustrates how the runtime execution times vary with the number of VIMs involved in the deployment options. The tag **'SS'** means that the `modeling` and/or `forecasting` functions are executed sequentially per resource type and per VIM, **'PS'** means that they are executed in parallel per resource type and sequentially per VIM, and **'PP'** means that they are executed in parallel per resource type and per VIM. Finally, simple accessing of the forecasted values is executed sequentially per VIM, which is tagged as **'S:get'**, where each execution takes around 42 μs on average, which has a negligible impact on the overall execution times: it can be observed how sequential execution results in a linear increase in the time measurements, growing from 12 to 119 s, while the execution in parallel remains constantly around 5 s regardless of the number of VIMs involved.

C. Placement Optimization

Reduced vApp graphs with $M = \{1, 2, \dots, 10\}$ macro nodes are considered in this evaluation. Particularly, all outputs from the **Reduce Graph** submodule for $N = 10$ (with varying number of links L, number of constraint metrics and threshold levels) have been sorted according to the resulting macro nodes.

For simplicity, but without loss of generality, we suppose that all of the original ten nodes have the same *flavor* (i.e., 1 vCPU,

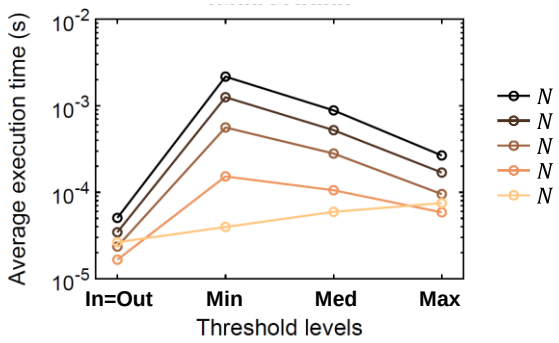
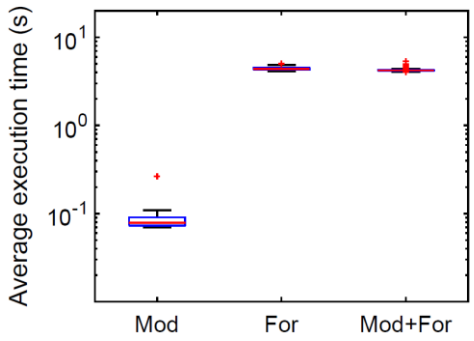
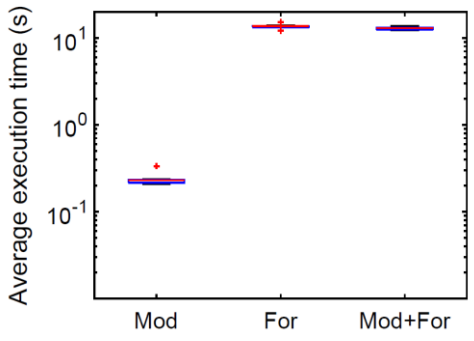


Fig. 5. Average execution times of the Graph Reduction submodule for varying input vApp graph sizes and threshold levels.



(a). per resource



(b). per VIM (sequential)

Fig. 6. Average execution times for building/updating the models and generating forecasts at the resource and VIM levels.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3151468, IEEE Transactions on Network and Service Management

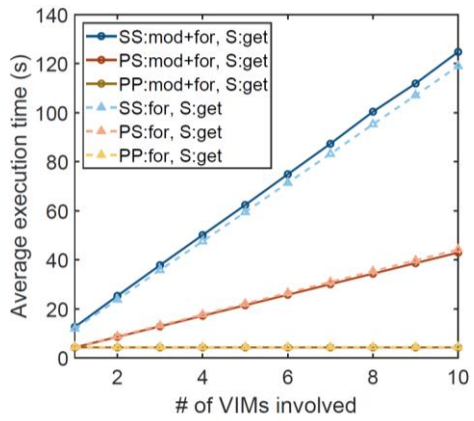


Fig. 7. Average execution times for runtime modeling and/or forecasting, for varying number of VIMs involved.

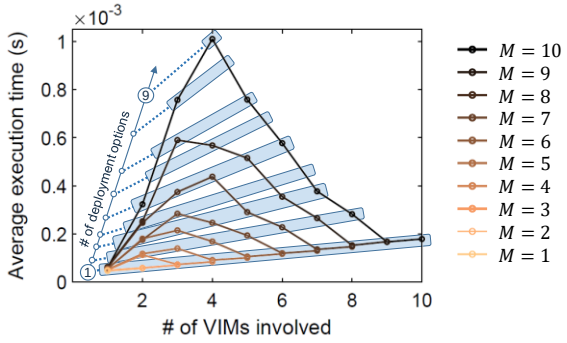
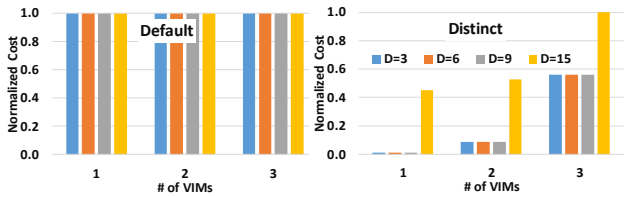
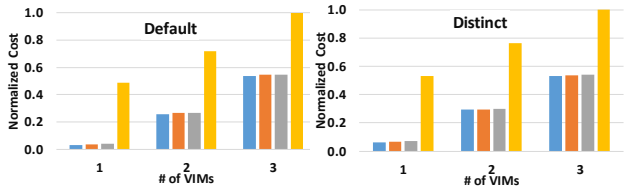


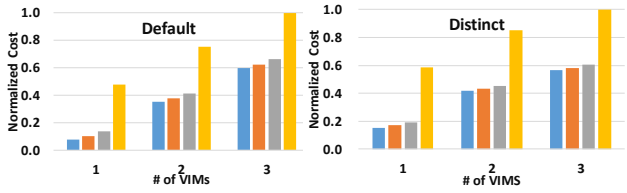
Fig. 8. Average execution times of the Placement Optimization submodule for varying number of macro nodes, VIMs involved and deployment options.



(a). $[\boldsymbol{Acost_1}, \boldsymbol{Acost_2}, \boldsymbol{Acost_3}] = [\mathbf{1}, \mathbf{1}, \mathbf{1}]$



(b). $[\boldsymbol{Acost_1}, \boldsymbol{Acost_2}, \boldsymbol{Acost_3}] = [\mathbf{5}, \mathbf{5}, \mathbf{1}]$



(c). $[\boldsymbol{Acost_1}, \boldsymbol{Acost_2}, \boldsymbol{Acost_3}] = [\mathbf{10}, \mathbf{5}, \mathbf{1}]$

Fig. 9. Comparing the deployment costs for varying number of VIMs involved, deployment options (D=3, 6, 9, 15), agreement costs and resource settings with *fixed* Υ'.

2048 MB of RAM, 20 GB of disk), such that the slice requirements ($Slice\_reqs$) can be easily derived based on the size of the macro nodes. Similarly, all the VIMs involved are supposed to have the same *default* resource information given as triples corresponding to (vCPU, RAM, disk): (a) free resources (50 vCPUs, 102400 MB, 100 GB), (b) usage (50%, 50%, 50%), and (c) overcommit ratios (16, 1.5, 1). These values are chosen such that each VIM has the capability to host even the entire slice. Since the WIM performance is out of the scope of this work, the network topology is irrelevant.

With the above considerations, the evaluation space for the deployment options is greatly reduced while still granting a meaningful analysis. In particular, Fig. 8 shows trends in the average execution times of the **Placement Optimization** submodule according to the number of macro nodes (M), VIMs involved (Υ') and deployment options (D). For a given M, it can be understood that D has a stronger influence than Υ' on the execution times. Moreover, while the results presented are obtained with the selection policy **P1**, it is interesting to note that similar trends are observed using **P2** and **P3**.

The placement optimization is then evaluated in terms of the deployment costs and execution times. Reduced vApp graphs with $M = \{1, \dots, 5\}$ macro nodes are considered; particularly, all outputs from the **Reduce Graph** submodule for $N = 5$ (with varying number of logical links L, number of constraint metrics and threshold levels) have been sorted according to the resulting macro nodes. Further, the number of candidate VIMs is limited to $\Upsilon = 3$ to better understand the impact of other deployment parameters, such as the agreement costs ($Acost_v$, $v = 1, 2, 3$), the resources' usage and overcommit ratios, the deployment options, and whether the parameter Υ' is considered as a *fixed* value (e.g., if $\Upsilon' = 3$, each deployment option should have three different VIMs involved), or as a *maximum* value (e.g., if $\Upsilon' = 3$, deployment options can either have one / two / three VIM(s) involved) in generating the options. In more detail, the VIMs can assume the agreement costs $[Acost_1, Acost_2, Acost_3] \in \{[1,1,1], [5,5,1], [10,5,1]\}$; then the resource usage and overcommit ratios are chosen according to the two settings below:

***S1.* Default**

All VIMs adopt the default values for (vCPU, RAM, disk) usage (50%, 50%, 50%) and overcommit ratios (16, 1.5, 1).

***S2.* Distinct**

Each of the three VIMs adopts distinct values for the (vCPU, RAM, disk) usage – (30%, 30%, 30%), (50%, 50%, 50%), (80%, 80%, 80%), and overcommit ratios – (16, 1.5, 1), (8, 1.3, 1), (1, 1, 1), respectively.

1) Deployment Costs

When the number of VIMs involved in each deployment option is strictly equal to Υ', Fig. 9 illustrates how the cost of the deployment (normalized to the maximum value) varies with the default / distinct resource settings and agreement costs. As for Fig. 8, the selection policies are not reported as their impact is not significant with both *Default* and *Distinct* resource settings as the agreement costs are varied. In the former, the cost difference between having 1 through 3 VIMs in the deployment increases with the agreement costs. This may be attributed to the interaction between the two weighting coefficients (α_v and $Acost_v$) included in the policies. Note that default resource settings will give the same values for α_v; in that case the VIMs are only differentiated according to the $Acost_v$. Apparently, the *Distinct* setting gives slightly higher costs, but this behavior is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3151468, IEEE Transactions on Network and Service Management

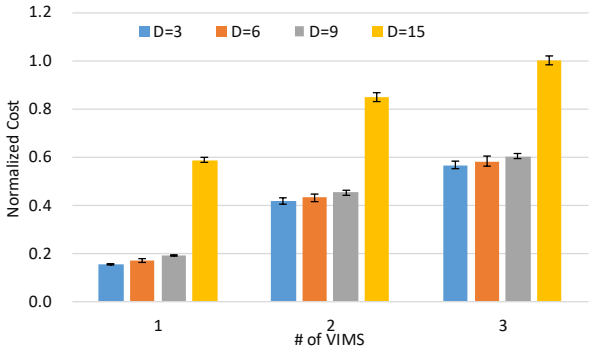
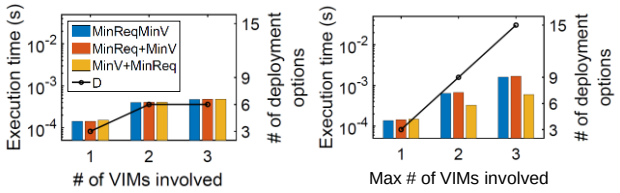


Fig. 10. Error bars representing the distance between deployment costs for the *Distinct* setting c) obtained by using forecasted data and measured ones.



(a). Fixed Υ' (b). Υ' as *maximum*

Fig. 11. Comparing the average execution times of the three selection

inversely proportional to the number of VIMs, and in fact costs are the same for 3 VIMs. For both settings, costs grow with the number of VIMs but this growth is quicker when $Acost$ is lower. Regarding the deployment options, they have a stronger impact as $Acost$ grows. The cost has a spike for D=15, which, in addition to the following considerations on the execution time, must be taken into account when interacting with the WIMs.

Figure 10 reports the same results as in Fig. 9, *Distinct* setting, $[Acost_1, Acost_2, Acost_3] = [10,5,1]$, but it also reports the error in the deployment cost caused by using the forecasted amount of free resources and usages instead of the measured ones. For all values of involved VIMs and deployment options, the error stays below 5%.

2) Execution Times

As regards the execution times of the three selection policies, the data further support the initial analysis of the results shown in Fig. 8– that is, the execution time is mostly influenced by the number of deployment options (D). Particularly, the variations in the resource settings and agreement costs did not affect much the execution times.

Fig. 11 shows the average execution times of the selection policies with Υ' fixed and as a maximum, also indicating the number of deployment options for both cases plotted as a black line in the figure. When Υ' is fixed, the policies run for similar durations; whereas when Υ' is considered as a maximum, the MinV+MinReq policy results in shorter durations by around 40~50% as Υ' (and D) increases. Note that the deployment option included for Υ' in the second case is simply the union of all deployment options for $\leq \Upsilon'$ in the first case (i.e., Fig. 11b indicates 9 (=3+6, from Fig. 11a) for $\Upsilon' = 2$, and 15 (=3+6+6, from Fig. 11a) for $\Upsilon' = 3$). It can be noticed that, even when Υ' is considered as a maximum and the execution time grows with the number of deployment options, such growth is far from being linear with D and the effect on the execution time is limited.

Based on the results obtained in both the deployment cost and execution time evaluations, it can be deduced that the **MinV+MinReq** policy presents a promising compromise.

Among the works reported in Section II, [20] presents a deployment algorithm that can help draw some considerations on the effectiveness of our work. Although the referred study proposes an evaluation of the execution time for multiple slices, it is apparent that the order of magnitude is in the range of the seconds. As shown in this section and corroborated by the overall evaluation performed in [26], the contribution of the three RSO submodules stays in the ms range making it a valuable asset for a realistic deployment.

VI. Conclusions

Emerging network softwarization solutions such as the MEC, NFV and SDN paradigms are key enablers of 5G technologies, as well as towards telecommunications infrastructure convergence and a distributed, multi-service ecosystem. In this scenario, multi-tenancy and the as-a-Service concept will also advance in the end-to-end network slicing direction, enabling vertical industries and network service providers to access/manage their assets via slice abstractions. The challenge remains in the instantiation and management of application-aware network slices necessary to support the multi-site deployment of vApps.

This paper presented the design and operation details of the RSO – a software-service in the MATILDA OSS, whose main goal is to select the best deployment among a list of options provided by the WIM. It consists of the Graph Reduction, Utilization Forecasting and Placement Optimization submodules, which respectively handle the aggregation of vApp components based on affinities, the forecasting of µDC resources utilization, and the optimization of the QoS-aware, multi-site deployment of the (aggregated) vApp components.

The RSO's performance is mainly evaluated in terms of the execution times of its submodules, while varying their input parameters, such as the vApp graph topology, constraint metrics, execution method, slice requirements, VIM resources, deployment options, etc. For the Placement Optimization submodule, three selection policies are also compared in terms of both execution times and deployment costs. Experimental results allowed to assess the suitability of the RSO to be a relevant asset for the upcoming 5G ecosystems. Moreover, results identified a number of different behaviors and trade-offs that can be exploited in engineering its inputs to improve the interactions with other OSS submodules and network platform components, not only for multi-site vApp deployment but also for other network/services management operations, while applying the most appropriate setup to the current environment and requirements.

Furthermore, it is worth noting that the highly flexible, microservice-based design of the RSO allows for seamless integration of other next-generation forecasting and optimization algorithms.

REFERENCES

[1] C. Tao *et al.*, "Vision on Software Networks and 5G," 5GPPP SN WG White Paper. Jan. 2017. [Online]. Available: https://5gppp.eu/wp-content/uploads/2014/02/5GPPP_SoftNets_WG_whitepaper_v20.pdf.

[2] "Mobile Edge Computing (MEC); Framework and Reference Architecture," ETSI GS MEC 003 v1.1.1, Mar. 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf.

[3] "Setting the Scene for 5G: Opportunities & Challenges," ITU Rep., 2018. [Online]. Available: https: //www.itu.int/en/ITU-D/Documents/ITU_5G_REPORT-2018.pdf.

[4] M. Chiosi *et al.*, "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges and Call For Action," ETSI White Paper. Oct. 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf.

[5] D. Kreutz, *et al.*, "Software-Defined Networking: A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[6] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94-100, May 2017.

[7] "5G for Business: a 2030 Market Compass," Ericsson Rep., Oct. 2019. [Online]. Available: https://www.ericsson.com/4a8e35/assets/local/5g/the-5g-for-business-a-2030-compass-report-2019.pdf.

[8] A. Zafeiropoulos *et al.*, "Enabling Vertical Industries Adoption of 5G Technologies: A Cartography of Evolving Solutions," in Proc. 2018 European Conf. Netw. Commun. (EuCNC), Ljubljana, Slovenia, June 2018, pp. 1-9.

[9] Q. Duan, S. Wang and N. Ansari, "Convergence of Networking and Cloud/Edge Computing: Status, Challenges, and Opportunities," *IEEE Netw.*, doi: 10.1109/MNET.011.2000089.

[10] "MATILDA – A Holistic, Innovative Framework for Design, Development and Orchestration of 5G-ready Applications and Network Services over Sliced Programmable Infrastructure." [Online]. Available: http://www.matilda-5g.eu/.

[11] "5G-INDUCE – Open cooperative 5G experimentation platforms for the industrial sector NetApps." [Online]. Available: https://www.5g-induce.eu.

[12] L. Ma, X. Wen, L. Wang, Z. Lu and R. Knopp, "An SDN/NFV Based Framework for Management and Deployment of Service Based 5G Core Network," *China Commun.*, vol. 15, no. 10, pp. 86-98, Oct. 2018.

[13] H. Chien, Y. Lin, C. Lai and C. Wang, "End-to-End Slicing as a Service with Computing and Communication Resource Allocation for Multi-Tenant 5G Systems," in *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 104-112, Oct. 2019.

[14] A. Kammoun, N. Tabbane, G. Diaz, A. Dandoush and N. Achir, "End-to-End Efficient Heuristic Algorithm for 5G Network Slicing," in Proc. IEEE 32nd Int. Conf. Adv. Inform. Netw. Appl. (AINA), Krakow, Poland, May 2018, pp. 386-392.

[15] W. Guan, X. Wen, L. Wang, Z. Lu and Y. Shen, "A Service-Oriented Deployment Policy of End-to-End Network Slicing Based on Complex Network Theory," *IEEE Access*, vol. 6, pp. 19691-19701, 2018.

[16] H. Halabian, "Distributed Resource Allocation Optimization in 5G Virtualized Networks," *IEEE J. Select. Areas Commun.*, vol. 37, no. 3, pp. 627-642, Mar. 2019.

[17] R. A. Addad, M. Bagaa, T. Taleb, D. L. C. Dutra and H. Flinck, "Optimization Model for Cross-Domain Network Slices in 5G Networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1156-1169, May 2020.

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

[19] X. Feng, Z. Lu, L. Wang and W. Guan, "A Delay-Aware Deployment Policy for End-to-End 5G Network Slicing," in Proc. 2019 IEEE Int. Conf. Commun. (ICC), Shanghai, China, May 2019, pp. 1-6.

[20] K. Ludwig, A. Fendt and B. Bauer, "An Efficient Online Heuristic for Mobile Network Slice Embedding," in Proc. 23rd Conf. Innov. Clouds, Internet and Netw. and Workshops (ICIN), Paris, France, Feb. 2020, pp. 139-143.

[21] Z. Yan, J. Ge, Y. Wu, L. Li and T. Li, "Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks," *IEEE J. Select. Areas Commun.*, vol. 38, no. 6, pp. 1040-1057, June 2020.

[22] K. Guo, Y. Wang, X. Qiu, W. Li and A. Xiao, "Particle Swarm Optimization based Multi-domain Virtual Network Embedding," in Proc. 2015 IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM), Ottawa, ON, Canada, May 2015, pp. 798-801.

[23] X. Wang, Q. Chen and H. Qiu, "A Effective Two-step Strategy of Multi-domain Virtual Network Embedding in 5G Network Slicing," in Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC), Chengdu, China, Dec. 2017, pp. 1174-1179.

[24] D. Bhamare, *et al.*, "Optimal Virtual Network Function Placement in Multi-cloud Service Function Chaining Architecture," *Comput. Commun.*, vol. 102, 2017, pp. 1-16.

[25] R. Bruschi, F. Davoli, P. Lago and J. F. Pajo, "Move with Me: Scalably Keeping Virtual Objects Close to Users on the Move," in Proc. 2018 IEEE Int. Conf. Commun. (ICC), Kansas City, MO, 2018, pp. 1-6.

[26] R. Bruschi, F. Davoli, F. Diaz Bravo, C. Lombardo, S. Mangialardi, J. F. Pajo, "Validation of IaaS-based technologies for 5G-ready applications deployment", Proc. European Conf. on Networks and Commun. 2020 (EuCNC 2020), Dubrovnik, Croatia, June 2020.

[27] R. Bruschi, F. Davoli, C. Lombardo, J. F. Pajo, "Managing 5G Network Slicing and Edge Computing with the MATILDA Telecom Layer Platform", Computer Networks, vol. 194, pp. 1-14, April 2021.

[28] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis and S. Tilkov, "Microservices: The Journey So Far and Challenges Ahead," *IEEE Softw.*, vol. 35, no. 3, 2018, pp. 24–35.

[29] "5G; System Architecture for the 5G System," ETSI TS 123 501 v15.5.0, Apr. 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.05.00_60/ts_123501v150500p.pdf

[30] S. Thalanany and P. Hedman. "Description of Network Slicing Concept." NGMN Alliance Deliverable. 2016. [Online]. Available: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf.

[31] "Study on Management and Orchestration of Network Slicing for Next Generation Network." TR 28.801, 3GPP TSG Tech. Rep. 2018. [Online]. Available: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf.

[32] "Network Functions Virtualisation (NFV), Management and Orchestration," ETSI GS NFV-MAN 001 v1.1.1, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFVMAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[33] "Mobile Edge Computing (MEC); End to End Mobility Aspects," ETSI GR MEC 018 v1.1.1, Oct. 2017. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/018/01.01.01_60/gr_mec018v010101p.pdf.

[34] R. Bolla, R. Bruschi, F. Davoli, P. Gouvas, and A. Zafeiropoulos, "Mobile Edge Vertical Computing over 5G Network Sliced Infrastructures: an Insight into Integration Approaches", *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 78-84, July 2019.

[35] The Ericsson Network Manager, https://www.ericsson.com/en/portfolio/digital-services/automated-network-operations/network-management/network-manager.

[36] MongoDB, [Online]. Available: www.mongodb.com/

[37] Prometheus, [Online]. Available: https://prometheus.io/

[38] SageMath, [Online]. Available: http://www.sagemath.org/

[39] O. Trull, J. C. García-Díaz, A. Troncoso, "Initialization Methods for Multiple Seasonal Holt–Winters Forecasting Models," Mathematics, 8(2), 268, Feb. 2020, https://doi.org/10.3390/math8020268".

[40] OpenStack, "Compute Schedulers." [Online]. Available: https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html

Raffaele Bolla is currently Full Professor of Telecommunication Networks at DITEN- University of Genoa. He received the PhD degree in Telecommunications Engineering in 1994 from the University of Genoa, Italy. He has been Principal

Investigator in a number of national and international research projects, as well as CNIT/University reference person for many industrial projects. Among others, he has been the Principal Investigator of the FP7 Integrated Project ECONET (low Energy Consumption NETworks) and has been involved as leader of the CNIT/University research group in several FP7 and H2020 projects (e.g., ASTRID, MATILDA, INPUT, ARCADIA, ECONET, TREND, Intermedia). He is currently the reference person for international relationships and students'/teachers' mobility management of the Polytechnic School of Engineering and Architecture of the University of Genoa. He has co-authored over 200 scientific publications and has been a member of technical committees of major international conferences. He is also strongly involved in standardization activities in the area of telecommunication networks sustainability in ETSI and ITU-T SG 5.

Roberto Bruschi is Associate Professor of Telecommunication Networks at the University of Genoa, Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN). Roberto has been the Project Coordinator of the RIA INPUT project, funded by the H2020 Programme. He took part in the activities of many national and European projects (e.g., MIUR GreenNet as Principal Investigator (3% access rate), H2020 MATILDA, H2020 SPIDER, FP7 IP ECONET, etc.). In the ECONET project, he served as Coordination Project Manager, and in 5G-PPP MATILDA was leading the R&D activities on the 5G Network Platform. Roberto has co-authored over 130 papers in international journals, book chapters and international conference proceedings, and he was the recipient of two Best Paper Awards (at IEEE ICC 2009 and at the 3rd IEEE Internat. Workshop on Green Communications), one Runner-Up Best Paper (at IoT World Forum 2015), and one Best Demo (at IFIP/IEEE IM 2017) Awards. He has been invited to a number of scientific international conferences and seminars for talks, tutorials and panels (e.g., IEEE INFOCOM 2012, IEEE HPSR 2011, FUNEMS 2012, NoF 2019, etc.). He also chaired some scientific workshops (ITC EPIF 2014, TIWDC 2013, etc.). He is a Senior Member of the IEEE.

Franco Davoli is Professor Emeritus at the University of Genoa, Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN). His current research interests are in dynamic resource allocation in multiservice networks and in the Future Internet, wireless mobile 5G/6G and satellite networks, multimedia communications and services, and in flexible, programmable and energy-efficient networking. He has co-authored over 380 scientific publications in international journals, book chapters and conference proceedings. In 2004 and 2011 he was Visiting Erskine Fellow at the University of Canterbury, Christchurch, New Zealand. He has been Principal Investigator in a large number of projects and has served in several positions in the Italian National Consortium for Telecommunications (CNIT), an independent organization joining 37 universities all over Italy. He was co-founder and Head, for the term 2003–2004, of the CNIT National Laboratory for Multimedia Communications, Naples, Italy, and Vice-President of the CNIT Management Board for the term 2005–2007. He is currently the Head of the CNIT National Laboratory of Smart and Secure Networks (S2N), based in Genoa, Italy, and coordinator of the H2020 5G PPP 5G-INDUCE project. He is a Life Senior Member of the IEEE.

Chiara Lombardo received her Ph.D. in Electronics, Informatics, Robotics and Telecommunications Engineering at the University of Genoa (funded by NetLogic Microsystems, now owned by Broadcom) in 2014. Chiara has worked as a postdoc research fellow at the University of Genoa, Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN) for six years and currently works with the CNIT S2N National Laboratory, where she is involved in most of the research and technical activities. She is currently involved in the H2020 5G-INDUCE and SPIDER Projects and has previously worked in the H2020 Projects INPUT and TRIANGLE and in the FP7 Project ECONET. She has co-authored over 20 papers in international journals, book chapters and international conference proceedings. Her current research interests cover NFV, edge computing and 5G networks.

Jane Frances Pajo received the B.Sc. degree (cum laude) in electronics and communications engineering from the Mindanao State University-IIT, Philippines, in 2010, while the M.Sc. (cum laude) and Ph.D. (Europaeus) degrees in Telecommunications Engineering were obtained from the University of Genoa, Italy, in 2015 and 2019, respectively. She held a post-doctoral research fellowship with the TNT Laboratory at the DITEN Department of the latter institution. She also worked with the CNIT S2N National Laboratory, Genoa. She is currently with Telenor Research, Fornebu, Norway. Her research interests include applications of network softwarization technologies and artificial intelligence for network/service management and orchestration.