This is a postprint version of the following published document:

Sciancalepore, V., Yousaf, F.Z. y Costa Pérez, X. (2018). z-TORCH: An Automated NFV Orchestration and Monitoring Solution. *IEEE Transactions on Network and Service Management*, 15, (4), pp. 1292-1306.

DOI: 10.1109/TNSM.2018.2867827

z-TORCH: An Automated NFV Orchestration and Monitoring Solution

Vincenzo Sciancalepore, Faqir Zarrar Yousaf and Xavier Costa Perez
NEC Laboratories Europe
69115 Heidelberg, Germany
Email: {vincenzo.sciancalepore, zarrar.yousaf, xavier.costa}@neclab.eu

***Abstract*—Autonomous management and orchestration (MANO) of virtualized resources and services, especially in large-scale NFV environments, is a big challenge owing to the stringent delay and performance requirements expected of a variety of network services. The quality of decision (QoD) of a MANO system depends on the quality and timeliness of the information that it receives from the underlying monitoring system. The data generated by monitoring systems is a significant contributor to the network and processing load of MANO systems, impacting thus their performance. This raises a unique challenge: how to jointly optimize the QoD of MANO systems while at the same minimizing their monitoring loads at runtime? This is the main focus of this paper.**

In this context we propose a novel automated NFV orchestration solution called z-TORCH (zero Touch Orchestration) that jointly optimizes the orchestration and monitoring processes by exploiting machine learning techniques. The objective is to enhance the QoD of MANO systems achieving a near-optimal placement of VNFs at minimum monitoring costs.

I. Introduction

Network Function Virtualization (NFV) is widely being considered as one of the key enabling technologies for 5G networks. One of the main motivating factors behind NFV is to provide a technology that will enable the operators and service providers to provide and manage resources and services in an efficient and agile manner with reduced CAPEX and OPEX, reduced new service roll-out time and increased ROI.

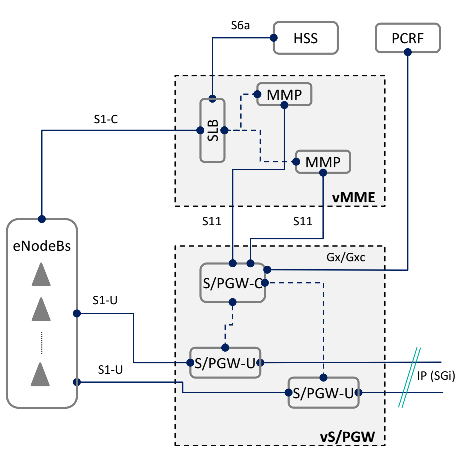


Fig. 1. Example of a vEPC VNF with its respective VNF Components.

An NFV system consists of Virtualized Network Functions (VNF) that are deployed on servers, commonly referred to as compute nodes, located inside the datacenter. A Cloud Management System (CMS) is an integral part of such an NFV Infrastructure (NFVI) that is responsible for the Management and Orchestration (MANO) of NFVI resources, such as compute nodes, CPU, network, memory, storage, VNFs etc. For effective MANO decisions a CMS relies on the presence of a reliable and robust monitoring system that monitors the utilization of the NFVI resources and VNF Key-Performance Indicators (KPIs) and keeps the CMS updated by the regular provisioning of monitored data and KPIs. The CMS will regularly analyze the monitored data and derive appropriate Lifecycle Management (LCM) decision. According to a conservative estimate, up to 25% of enterprise data today is from systems monitoring, with almost 240 terabytes produced annually [1]. This is likely to grow many folds with the wide deployment of NFV. *The challenge thus is to achieve optimum MANO decisions with reduced monitoring load.*

As part of the MANO operation, a CMS will impart relevant LCM actions on the individual VNFs and/or its underlying resources in order to ensure its operational/functional integrity. LCM actions may include scaling in/out/up/down, migration/placement, update/upgrade, delete etc. of individual VNFs and/or its respective resources. Arriving at the correct LCM decisions is by itself an incredible challenge owing to the variety of VNFs that needs to be managed inside an NFVI. The complexity of a VNF may also vary where more complex VNFs may embody a complete system, for example a virtualized EPC (vEPC) system that is formed of multiple VNF components (VNFC) interlinked over standard and proprietary virtual links. The example of such a complex VNF is illustrated in Fig. 1 [2]. The MANO complexity of a CMS further increases as it also needs to manage not only VNFs but also Network Services (NS) that are formed by chaining relevant VNFs, e.g. firewalls, video optimizers, schedulers, virtualized EPCs, etc.

If the LCM decisions on actions are not taken with care and deliberations, the LCM actions on one or more resource element(s) (e.g., infrastructure resources, VNFs etc.) may have an inadvertent adverse impact on other resource elements that may be relying on the services of the managed resource element and/or sharing the other resources. For example, a migration decision on a VNF belonging to a particular active NS may not only have an adverse impact on the overall QoS of the NS itself, but it may also inadvertently impact on the QoS of other VNF(s)/NS that may be sharing its resources with the migrated VNF due to resource contention. The QoS degradation of one NS may also impact on the QoS of other NS that were relying on the services offered by that particular

NS. The CMS will thus perform a second iteration of LCM actions to rectify from this degraded service situation, and thus it is very much likely for the CMS to run multiple iterations before a stable and optimum situation is achieved. However, it is highly undesirable to run multiple iterations of LCM/orchestration decisions within a short span of time as it results in continuous service interruption thereby impacting the overall Qos/QoE. In other words, the CMS will have a poor Quality of Decision (QoD).

The notion of QoD was first introduced in [3], which is an indicator of the effectiveness of CMS in terms of imparting MANO decisions. According to [3] the QoD is measured in terms of the following two mutually dependent criteria:

1) How resource efficient the management action is. The resource efficiency is in turn measured in terms of:
    - Whether both the long term and short term resource requirements of the managed VNF will be fulfilled in the selected compute node.
    - How non-intrusive a management action has been for other VNFs that are already provisioned in the selected compute node. That is, to what extent will the managed VNF VM affect the performance of other VNFs in the selected compute node in terms of resource availability.
2) Number of times the management action has to be executed before the most-suitable compute node is determined to migrate/scale the managed VNF to.

The QoD of the CMS in turn depends on both the *quality* and *quantity* of the information that it receives from the monitoring system. The quality depends on the variety of KPIs that is reported to the CMS while the quantity depends on the frequency of updates on the KPIs that the CMS retrieves. Information provided by a monitoring system may include a variety of KPIs, e.g., percentage-utilization of specific resource units and/or aggregate resource utilization values of all the VNFs in a physical machine, load experienced by individual VNFs, other QoS parameters etc. The CMS may then analyze the received data in order to find the state of NS and take appropriate LCM actions, for example, whenever it senses high load/utilization events. Moreover, a CMS may manage and orchestrate services that span across multiple data-centers that are geographically apart [4] and thus rely on receiving monitored data from all the data centers that is under the CMS administrative domain. However, the problem being that considering the size of an NFVI, where a single NFVI-PoP may host 100s of 1000s of compute nodes and, each compute node may host 10s of 100s of VNFs and thus host and manage 1000s of NS instances. The scale of the assets/resources that the CMS requires to monitor will further increase in case of multiple data-centers.

In consideration of the scale of the resources that the CMS must monitor will result in a very high load that must be delivered periodically by the monitoring system to the CMS and will also incur a high processing load as the CMS has to process and analyze all this data. This will also incur processing delay that may result in sluggish reaction to unwanted events by the CMS. Even with the provisioning of sufficient monitored data, the QoD of the CMS can not be guaranteed as it depends also on the intelligence of the orchestration algorithm of the CMS utilizing data from the monitoring system.

The challenge is thus to jointly optimize both the CMS orchestration process and monitoring process. In this paper we propose a novel orchestration mechanism, which we refer to as zero-Touch ORCHestration (z-TORCH) method that autonomously enhances the QoD of the CMS orchestration logic at minimum monitoring load at run-time. The challenge becomes all the more complex considering the multi-dimensional nature of the cloud infrastructure with a variety of KPIs and resources to consider resulting in a myriad of permutations. This challenge beiQ-learning ng the main objective of this paper, which we address by employing machine learning based method.

The contributions of our paper can be summarized as follows: i) we propose an unsupervised binding affinity process in order to profile the VNF KPIs, unveil the correlations between VNF behaviors and group them into VNF affinity groups, ii) we analytically study the complexity of our z-TORCH solution and empirically evaluate its convergence properties, iii) we devise an adaptive mechanism to dynamically change the number of affinity groups and properly tune its accuracy, iv) we adjust the CMS monitoring frequency based on VNF statistical information by means of Q-learning theory, v) we use a commercial virtualized EPC to configure our VNF profiling for performance evaluation purposes and vi) we show via an exhaustive simulations campaign that z-TORCH exhibits near-optimal performance at low monitoring costs.

The rest of the paper is organized as follows. The next Section II will give an overview of the related work. This is followed by Section III providing the detailed description of the system model and the algorithmic details of the novel z-TORCH method. Section IV shall provide the details of our simulation environment and the performance analysis of the proposed z-TORCH method. We also propose options for the practical deployment of our proposed method in a standard CMS, which is the ETSI NFV MANO system [5] in Section V. At the end we will present a summary of our work and analysis in Section VI.

II. Related Work

The work presented in this paper focuses on the joint optimization of VNF orchestration and monitoring process. In terms of monitoring process there are three main modes through which the CMS may receive monitored data:

1) *Periodic Mode* - Enables periodic delivery of monitored data, where the period and type of data is specified.
2) *Pull Mode* - Provides monitored data only when solicited by the CMS.
3) *Push Mode* - Sends monitored data only when a specific event is triggered, for example, CPU burden or a network load on a VNF exceeds some specific threshold.

While those methods, and combination of them [6], have been exhaustively explored in the literature, they present significant limitations. Periodic reports are identified as the straightforward approach to keep monitoring the resources status but, in case of very large data-centers, it considerably exacerbates the burden and complexity of the monitoring process. Conversely, pull requests option solves the huge overhead issue but it needs a proper design in order to provide the QoE/QoS guarantees and may make the CMS miss out on some critical events. Lastly, the push mode can be tuned so as to recover the system when it is close to alert-states but it may prevent from an optimal allocation/distribution of VNFs within the available compute nodes. Thus none of these three traditional techniques offer a reliable and optimized solution for large scale NFVI-PoPs and their shortcomings will thus have an adverse impact on the CMS' QoD. There is thus a need to develop an adaptive approach where the monitoring system can adapt according to the events.

In terms of adaptive monitoring systems, there are proposals related to adaptive sampling especially in the domain of wireless networks where energy, processing and bandwidth resources are at premium. Some of them utilize learning techniques like reinforcement learning to make optimum choice of sampling data. Typically such approaches would include clustering, data aggregation and prediction to determine the data sampling frequency. For example [7] proposes an adaptive model selection (AMS) algorithm that relies on a-priori knowledge of models which is used by the sensor to compare its real measurement with the predicted ones, and only communicate data in case of large variance between the measured and predicted values. This saves on the communication load but it is still computationally expensive as the sensors need to continually sample measurements besides other shortcomings. [8] optimizes the query method of GWs for collecting periodical data from the monitored objects by employing a statistical technique called principal component analysis on historical traces of sensory data to automatically identify sensors that measured most of the variance observed in the environment. Data from only those sensors would then be collected reducing the transmission cost by up to 50%. This approach however does not take into account unpredictable environmental evolution yielding inaccurate data. Such a method is not feasible owing to the more frequent unpredictable workload variation on VNFs inside the NFVI. Another proposal is [9] that employs single rule defining the sampling interval according to the Time of Day, where sampling frequency is high during busy hours periods. It also employs Dual Prediction scheme (DPS) for prediction outside the busy hour based on historical data. This method again cannot be relied on in large scale NFV environment where multiple NS may exist with a different busy hour definition. A more recent work reported in [10] proposes a dynamic sampling rate adaptation scheme based on Reinforcement Learning that is able to tune temperature sensors sampling interval on-the-fly, according to environmental conditions and application requirements. The optimization goal is to avoid oversampling and save energy. The method selects from a predefined set of sampling frequencies making it unsuitable for the more dynamic and multi-variable NVF environment. Moreover, adaptive sampling methods usually focus on intelligently varying sampling frequency but ignore the duration of the surveillance epoch. Both these factors are crucial in NFV environment as the CMS is supposed to consider a LCM decision at the end of each surveillance epoch.

In the context of NFV orchestration, a large library of work is present that proposes different VNF placement algorithms with different optimization goals. Each proposed solution is unique to its own problem space and use case. We only present some of the more recent works in order to give an overview of the prevailing trends and needs of the industry in this very important problem space.

The authors in [11] propose VNF placement algorithms with two-fold objective of minimizing path between users and data anchor gateways, and optimizing the sessions' mobility. In [12] the authors propose a time-efficient heuristic based on affiliation-aware VNF placement for NS deployment. It also proposes an on-line forecast-assisted NS deployment algorithm that predicts the future VNF requirements. For optimizing the VNF placement decisions in response to on-demand workload, [13] proposes a solution called Simple Lazy Facility Location (SLFL) that results in the doubling of workload acceptance while incurring similar operational costs compared to first-fit and random placements. [14] explores the problem of VNF placement problem in the context of network load balancing in data centers. It explores the placement of VNFs in smaller clusters of servers in the network thus minimizing the distance-to-the-datacenter problem while considering the resources utilization. The authors study the problem of VNF placement with replications to help load balance the network. They design and compare three optimization methods, including Linear Programing (LP) model, Genetic Algorithm (GA) and Random Fit Placement Algorithm (RFPA) for the allocation and replication of VNFs showing significant improvement in load balancing. In the context of enterprise WLAN, [15] proposes a VNF placement algorithm for optimizing the functions deployment according to application level constraints. The proposal depends on the presence of hybrid nodes that combine the forwarding capabilities of a programmable switch with the storage/computational capabilities of a server. On similar trends, [16] studies the on-demand deployment of VNFs in telco CDNs.

All of the above cited work tackles the VNF placement problem with a narrow viewpoint of a particular use case with specific requirements. However there is a need to have a more universal approach to the VNF placement problem in particular and NFV Orchestration in general. Moreover, none of the above proposals take into account the orchestration cost and/or the monitoring load. The only work that does consider the CMS orchestrator QoD is one of our earlier works in [3], [17], in which we present a Resource Aware VNF Agnostic (RAVA) orchestration method that employs a very different approach of using Pearson Correlation method for optimum placement of VNFs with reduced orchestration cost. This method relied heavily on the frequent provisioning of monitored data from the underlying monitoring system. Moreover, the method did not provide an accurate VNF profile which is so crucial for the placement decisions. It is in view of this that we propose z-TORCH that jointly optimizes the NFV orchestration and monitoring process such that we achieve near-optimal placement of VNFs at reduced monitored load while enhancing the CMS QoD.

III. System concepts and model

Let us consider a generic cloud system shown in Figure 2. The infrastructure consists of multiple servers (referred to as compute nodes) and VMs deployed in each server. The server resources (e.g., compute, network, storage, memory) are virtualized and allocated to each VM based on the respective VM requirements. A VM when configured with some network function is referred to as a VNF, or when configured with

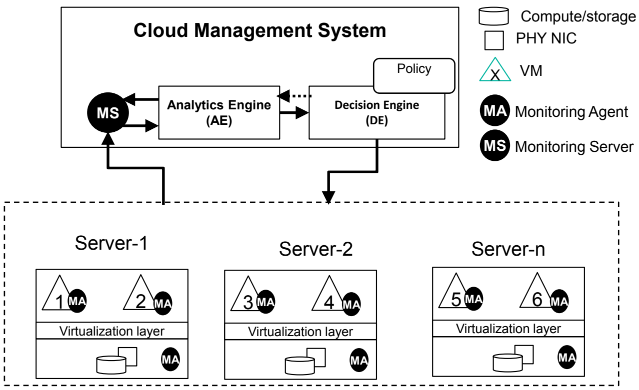


Fig. 2. Generic cloud management system.

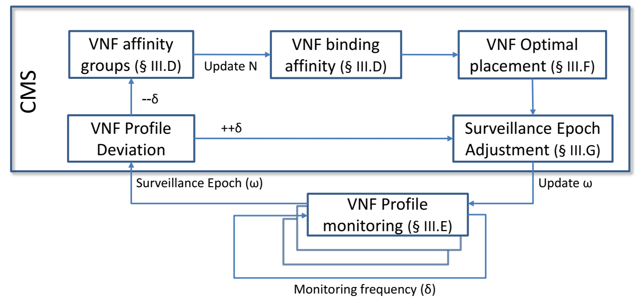


Fig. 3. Functional blocks of the z-TORCH solution.

some application function it is referred to as a Virtualized Application Function (VAF) [18]. For the sake of clarity we consider only VNFs. A VNF may belong to one or more virtual service instance and the CMS is supposed to manage and orchestrate the infrastructure resources in order to ensure service integrity and to ensure that each VNF is able to fulfil the operational/functional needs of the respective application/function configured inside it. To do that the CMS relies on an advanced monitoring system. There are many options available of how a monitoring system can be implemented and integrated, however in our work we consider a monitoring system (such as Zabbix [19]) where the Monitoring Server (MS) is deployed and configured within the CMS, and the MS interacts with one or more Monitoring Clients (MC). The MC instances are distributed within the infrastructure and each MC instance is associated with the entity, for example a VNF and/or a compute node, that needs to be monitored. A MC is an agent for the MS that simply monitors, samples, collects and record the relevant metrics and provides them to the MS. The MS after necessary pre-processing passes the data to the Analytics Engine (AE) that shall process the monitored data and provide the required analysis output to the Decision Engine (DE). The DE then takes some relevant LCM decisions based on some prescribed policy. Please note that the DE informs the the AE with its decision choice and, based on this, the AE is able to derive and provide suitable configuration parameters to the MS for future monitoring rounds.

The MC is configured via the MS by specifying the relevant metrics and/or KPIs that the MC is supposed to monitor and record. The MC is also configured with the monitoring granularity by specifying the time periods at which it should monitor, sample and record the specified metrics/KPIs.

A. z-TORCH: A dynamic monitoring and deciding process

Our proposal allows to re-think the classical CMS monitoring and function placement process by introducing machine-learning concepts. In particular, we devise a new solution able to (i) properly select and monitor relevant VNF KPIs, (ii) evaluating them based on prior (learned) information, (iii) optimally place them into available compute nodes to keep the system within stable working conditions, (iv) derive and schedule the next monitoring/decision times based on VNFs behaviour prediction. While this entire framework might appear complex and over-demanding, however it distributes the complexity of a centralized solution over MCs that are dynamically configured. Fig 4 shows the overall process. We define a *Surveillance Epoch* (ω) as the time window within which we monitor the VNFs' KPIs. Monitoring operations are performed at different *Sample points*, spaced by δ. Surveillance epochs last ω and are delimited by *decisional points* defined as points in time where our solution takes LCM decisions. In particular, LCM decisions might comprise: i) changing the frequency of monitoring information (δ), ii) changing the length of the surveillance epoch (ω), iii) optimal placement of VNFs based on *unsupervised binding affinity* calculation. When an alert message is captured by a sample point, the decisional point may be expedited to handle unexpected network changes.

B. General solution overview

Our novel concept of self-monitoring and proactive function placement relies on the concept of an adjustable monitoring frequency based on the machine-learning paradigm. Fig. 3 provides an example of the general process, which also indicates the relevant sections where the respective process item is described. At the beginning the decisional point requires the CMS to generally place VNFs onto available compute nodes. This initial operation might be performed without any a-priori information, namely *blind-placement*, or with some previous information gathered during a training phase. After placing VNFs, the CMS decides the frequency of sample points, i.e., the frequency of monitoring requests each server has to feed back to the CMS. This directly affects the overall monitoring load that might be unaffordable when facing with thousands of VNFs [1]. In addition, the CMS decides the length of the surveillance epoch based on a reward function obtained by a load forecasting module.

The KPIs of any single VNF are identified, based on VNF descriptors available beforehand [5], and are processed. This helps to provide an accurate *profile* of each VNF running in our system, as described in Section III-C. While the number of KPIs may consistently grow, our solution proposes an unsupervised binding affinity calculation to properly find out the correlation among them for any specified VNF. For the sake of simplicity, we consider the generic KPIs for any VNF, such as *Network Load*, *Computational Burden* and *Storage Utilization* [1]. A clear example is represented by a firewall VNF. It might be characterized by a high network demand and high storage utilization whereas it might exhibit low computational burden. Affinity values, which indicate the

[1]While the number of KPIs might be consistent, our solution still provides reasonable results when compared against state-of-the-art solution, as shown in the next sections.

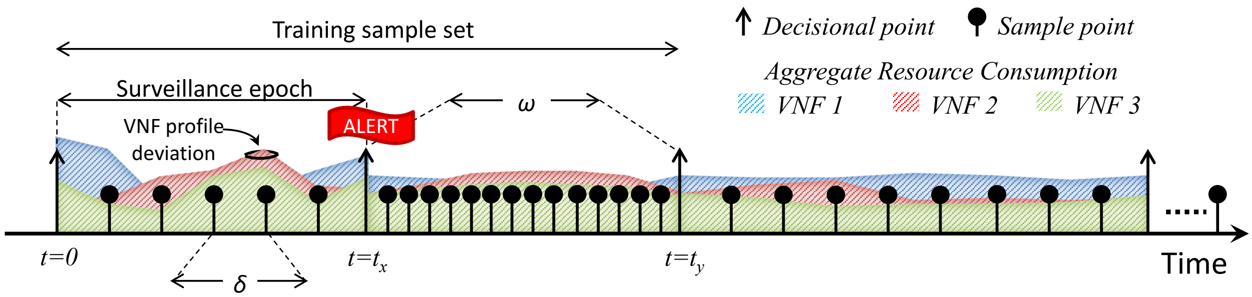


Fig. 4. Time evolution of z-TORCH.

correlation among different VNF profiles, are gathered by the CMS, which can optimally place the VNFs into compute nodes while keeping the overall load of any single compute node in balance. When the functions placement occurs (at $t = 0$ in Fig. 4), a default monitoring frequency δ and the surveillance epoch ω are fixed. At the next decisional point (ω), the CMS will detect any VNF differing from the prior profile information, namely *VNF profile deviation*. This automatically forces the CMS to increase the monitoring frequency in order to anticipate any unexpected critical event, such as compute node resources outage. At the next decisional point, if no other VNF profile deviation has occurred, the monitoring frequency is reduced and the reward function is increased (as explained in the Section III-G), which, in turn, enlarges the surveillance epoch ω. Conversely, if additional VNF profile deviations have occurred, a new VNF profiling is performed based on the *Training Sample Set*. In this case, the monitoring frequency is restored to a default value and the Surveillance epoch length is reduced (as the reward function is decreased).

C. VNF profiling process

VNF characteristics can be efficiently analysed with the aim of properly profiling the resource utilization. In particular, we rely on machine learning techniques in order to deeply learn from general behaviours to predict future trends and to be proactive in case of compute node resources shortages.

We can define the vector $\boldsymbol{p}_i^{(t)} = \{m_i^{(t)}, \mu_i^{(t)}, \eta_i^{(t)}\}$, where $m_i^{(t)}$ as the set of monitored information for each VNF i running in our system at time t whereas m, μ, η are the utilization of storage, CPU and network resources, respectively. This vector can be depicted as a single-point in a 3-dimensional space $\mathbb{S}$ within a snapshot of time t. In Fig. 5, we show different VNF profiles at consecutive time-snapshots to give a clear idea of our model. The plotting space can be partitioned to identify zones with specific profile properties. For instance, we have highlighted with a yellow sphere the zone wherein VNFs are marked as high-demanding: the main idea is to leverage on such profile partitioning in order to proactively place VNFs into compute nodes while keeping the system stable, i.e., when it does not requires further VNF migrations.

D. Unsupervised binding affinity

After defining our modelling space $\mathbb{S}$, we need to characterize different areas based on some peculiarities of all gathered

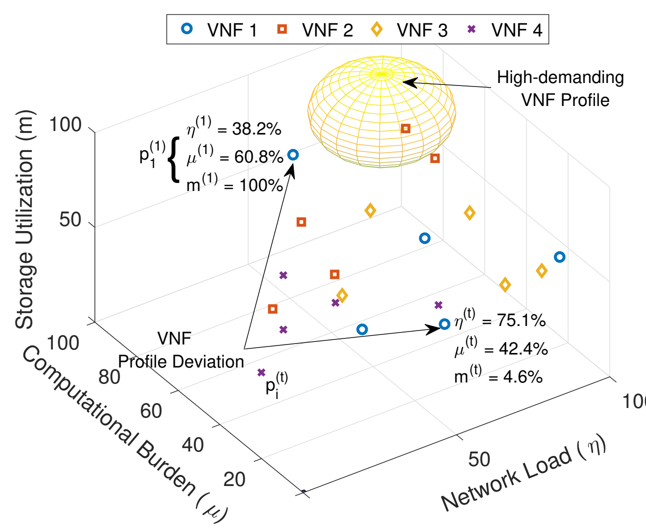


Fig. 5. Characterization of VNF profiles in a 3-dimensional space.

VNF profiles. Without loss of generality, we truncate the index (t) from $p_i^{(t)}$ when not needed. Given a number of affinity profile groups $N = |\mathcal{N}|$, our problem can be formalized as the following: *Finding non-overlapping affinity profile groups $n \in \mathcal{N}$ such that i) the union set of those groups is equal to the VNF profile space $\mathbb{S}$, ii) each affinity group contains at least one element $\boldsymbol{p_i}$, iii) all VNF profiles p_i must be placed in one VNF profile group.*

Each VNF profile group $n \in \mathcal{N}$ is characterized by a center of gravity $\boldsymbol{c_n} = \{c_{n,(1)}, c_{n,(2)}, \cdots, c_{n,(z)}\}$, where $z \in \mathcal{Z}$ is the spatial dimension ($Z = |\mathcal{Z}| = 3$ in our example). The center of gravity of group n is obtained as the spatial point with the least Euclidean distance from all other VNF profile values p_i associated to that group n. Mathematically, we can formulate the optimization problem as the following

Problem `VNF-Affinity`:

$$
\begin{aligned}
\text{minimize} \quad & \sum_n^{|\mathcal{N}|} \sum_i^{|\mathcal{I}|} x_{i,n} (||\boldsymbol{c_n} - \boldsymbol{p_i}||_2) \\
\text{subject to} \quad & \sum_n x_{i,n} = 1, \quad \forall i \in \mathcal{I}; \\
& \sum_i x_{i,n} \geq 1, \quad \forall n \in \mathcal{N}; \\
& \boldsymbol{c_n} \in \mathbb{R}^{|\mathcal{Z}|}; \\
& x_{i,n} \in \{0, 1\},
\end{aligned}
$$

where the outputs are $\boldsymbol{c_n}$ defining the spatial coordinates (KPIs) of each center of gravity and the binary values $x_{i,n}$ indicating whether VNF i is grouped into affinity group n, whereas $||\cdot||_2$ is the Euclidean distance between the center of gravity $\boldsymbol{c_n}$ for affinity group n and each VNF profile p_i. Specifically, the Euclidean distance depends on the number of KPIs (or spatial dimensions Z) considered, i.e., $||\boldsymbol{c_n}-\boldsymbol{p_i}||_2 = \sqrt{\sum_z^Z \left(c_{n,(z)} - p_{i,(z)}\right)^2}$. In our example, it holds that $p_{i,(1)} = m_i, p_{i,(2)} = \mu_i, p_{i,(3)} = \eta_i$.

In the preceding paragraphs we perform the complexity analysis and explain how our heuristic ekm works. Then we describe the process of calculating the density of the affinity groups N based on the current system status. Please note that the number of affinity groups N is decided beforehand and provided to our heuristic. This shall allow for the CMS to automatically cope with unexpected system changes and quickly react to keep the system stable.

Complexity analisys. While the number of affinity groups is given, Problem `VNF-Affinity` might be still untractable and it might not be solved in an affordable time. This is stated in the following theorem.

Lemma 1. *Given a number of affinity profile groups $N \geq 2$, multiple VNF $i \geq N$ and multiple KPIs $Z \geq 2$, Problem `VNF-Affinity` is NP-Hard.*

Sketch of Proof: We consider $Z = 2$ KPIs and $N = 2$ affinity profile groups. It is clear that the problem falls in NP. We can apply a polynomial reduction to the well-know graph k-coloring problem ([20]). In particular we are given an instance of the graph $G(V, E)$ wherein vertices are VNF profiles $V = \{1, 2, i, \ldots, I\}$ and edges are placed between two points with the largest distance. Therefore, we can formulate the following problem: is there any graph coloring solution that assigns different colors to vertices connected with the same edge? Assuming that this problem is NP-Complete and considering the color of each vertex as an affinity group, we can state that this problem is reducible to Problem `VNF-Affinity` in a polynomial time and thus, Problem `VNF-Affinity` is NP-Hard. When considering multiple affinity profile groups $N \geq 2$, it is hard to place the edges in the k-coloring graph [21], making the problem even harder. When considering multiple affinity groups and multiple KPIs $Z \geq 2$, it is even more difficult to find a solution to Problem `VNF-Affinity`, which proves that the NP-Hardness is rather strong.

□

Enhanced K-means heuristic. When dealing with NP-Hard problem, a fast and reasonable heuristic is needed to boil down the complexity of a greedy-search solution. There is a large library of work that address this problem, but we focus on a k-means heuristic solving the problem within $O(I \cdot N \cdot Z \cdot c)$ time-complexity [22], where c is the number of rounds to converge as explained next.

We rely on the classical definition of k-means algorithm and improve it to handle the complexity of our VNF affinity modelling. The main idea behind the well-known algorithm is to devise an iterative-algorithm able to randomly select the centres of gravity $\boldsymbol{c_n}$ (regardless of the number of spatial dimensions Z) and partition the whole space based on the nearest distance rule from each of those centres $\boldsymbol{c_n}$. Iteratively, the algorithm recomputes the new centres of gravity based on the current group member properties $\boldsymbol{p_i}$ and apply again the partitioning process until the centres of gravity do no change their positions. As proved by [22], in the worst-case the algorithm might take up to $2^{\Omega(\sqrt{I})}$ steps to converge. We enhance the performance of such an algorithm by applying a regular grid on the affinity space $\mathbb{S}$, namely enhanced k-means (ekm) algorithm. Points $w_s \in \mathcal{S}$ of the grid are equally spaced from each other. We then constrain the centres of gravity of each VNF affinity group to reside on some specific spatial points. The granularity of such grid span, i.e., the distance Δ, drives the speed of our algorithm and may be dynamically adjusted to speed up the process while keeping the accuracy of the found solution. This is performed by a step-function $\Delta^{(t)} = \frac{100}{2^{|\mathcal{I}|}} \cdot t^{\sqrt{I}}$: the more the steps to converge, the higher the slope of the step-function. Practically, we have designed a step-function which grows slowly during the first steps (depending on the number of VNF profile points I, i.e., the more the points, the slower the growth) and then, it exponentially grows as the number of steps becomes consistent. This helps the system to find a very accurate solution in the first steps, while forcing the algorithm to quickly converge if the number of steps is high.

Algorithm 1 Enhanced k-means (ekm)

1) Initialise $t = 0$ and $x_{i,n} = 0, \forall i \in \mathcal{I}, n \in \mathcal{N}$.
2) Initialise set $\boldsymbol{c_n}^{(t)}$ by using the VNF profiles classification.
3) Update $\Delta^{(t)} = \frac{100}{2^{|\mathcal{I}|}} \cdot t^{\sqrt{I}}$.
4) Apply a grid of points $\boldsymbol{w_s} \in \mathcal{S}$ on the VNF profile space $\mathbb{S}$ such that $||\boldsymbol{w_\xi} - \boldsymbol{w_\zeta}||_2 = \Delta^{(t)}, \forall \xi \neq \zeta, (\xi, \zeta) \in \mathcal{S}$.
5) Set $x_{i,n}^{(t)} = 1 : n = \arg\min_n ||\boldsymbol{c_n}^{(t)} - \boldsymbol{p_i}||_2, \forall i \in \mathcal{I}$.
6) Calculate the center of gravity set $\boldsymbol{c_n}^{(t+1)} = \frac{1}{\sum_i x_{i,n}^{(t)}} \sum_{i \in \mathcal{I}} \left(\boldsymbol{p_i} \cdot x_{i,n}^{(t)}\right), \quad \forall n \in \mathcal{N}$.
7) Update the center of gravity set based on the nearest grid point $\boldsymbol{c_n}^{(t+1)} = \boldsymbol{w_s} : s = \arg\min_{s \in \mathcal{S}} ||\boldsymbol{w_s} - \boldsymbol{c_n}||_2, \forall n \in \mathcal{N}$.
8) If $\boldsymbol{c_n}^{(t+1)} \neq \boldsymbol{c_n}^{(t)}$ then increase $t = t + 1$ and jump to (3).

The algorithm pseudo-code is provided in Alg. 1. To avoid the effects of randomness and to make our solution more efficient, we initialize the set of centres of gravity $\boldsymbol{c_n}$ (line 2) based on a VNF profile classification. Interestingly, this classification can be performed by means of external information providing VNF profile templates (in terms of expected KPIs), given a number of VNF affinity groups N. For example, when $N = 2$ VNF affinity groups are defined, VNF profile templates may influence the initial choice by placing the centres of gravity at $\boldsymbol{c_{n=1}} = \{75\%, 75\%, 75\%\}$ for the high-demanding VNF profiles and at $\boldsymbol{c_{n=2}} = \{25\%, 25\%, 25\%\}$ for the low-demanding VNF profiles. Clearly, such training data may be automatically updated by the infrastructure provider through a monitoring process.

VNF affinity groups density. While ekm algorithm can solve and provide a VNF affinity grouping solution within an affordable time, the key-aspect is the number of VNF affinity groups to build. We leverage on the feedback-loop paradigm to design a *controller* in charge of monitoring the system status and triggering a different number of VNF affinity groups when some events occur. The rationale behind this is

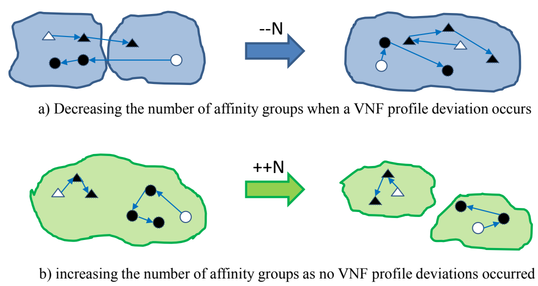


Fig. 6. Changing the number of VNF affinity groups based upon VNF profile deviation occurances.

that the affinity grouping procedure may fail and we need to promptly update the number of groups to handle unexpected VNF profile behaviors. Therefore, with some abuse of notation we define the concept of *VNF profile deviation*, as introduced in Section III-B, as follows:

Definition 1. *A VNF profile deviation is an event occurring when a VNF profile $p_i^{(t)}$ changes its KPIs from time t to $t+1$ falling into a new VNF affinity group $n \in \mathcal{N}$, i.e., $x_{i,n}^{(t)} \neq x_{i,n}^{(t+1)}$.*

VNF profile deviations give an indication about the accuracy of our affinity grouping process: if our grouping process failed to capture the variance of its members ($\boldsymbol{p_i}$), we need to re-run the affinity grouping process assessing the new VNF profile features. This may highly impact on the VNF function placement (as will be discussed in Section III-F) and may result in a service disruption because of a compute node resource outage.

Fig. 6 shows an example for a 2-dimensional space, i.e., considering only 2 KPIs for any VNF profile $\boldsymbol{p_i}$. In particular, we show a VNF profile at different times (with solid filled shapes). Please note that those values are snapshots captured at different *sample points*, as explained in Section III-B. When a VNF profile deviation occurs, an alert message is triggered and more sample points are required (in the next surveillance epoch) to take over the compute node control if some VNF profile exceeds the maximum capacity. At the next decision point, a new VNF affinity binding process is executed and the number of VNF affinity groups N is reduced. This will likely avoid further VNF profile deviations and perform an optimal VNF placement. Conversely, if the VNF profile behavior is predictable and does not exhibit significant changes, after 2 surveillance epochs the system automatically increases the number of VNF affinity groups to be more accurate in the VNF profiling process. We initially assume the lowest possible number of VNF affinity groups set to 2.

E. Learning and application

The number of sample points gathered for the VNF profiling process could significantly affect the VNF placement and, in turn, the overall network performance. Ideally, an infinite number of monitoring samples unveils the correct behavior of such VNF making accurate the VNF profiling. However, this might exacerbate the complexity and the overhead of control messages when applied to a plethora of VNF instances. In our proposal, we trade off the number of monitoring samples against the accuracy of VNF profiling process, which may lead to a huge number of LCM operations, such as migrations or scaling up/down.

Let us consider the realization of a point process $\gamma_{i,(z)} = \sum_{t=0}^{T} \delta_t p_{i,(z)}(t)$ as the evolution of VNF i and KPI (z), where δ_t is the Dirac measure for sample t.

Lemma 2. *VNF statistical properties can be obtained from any realization of the same process over time or from multiple process instances evaluated at the same time.*

Sketch of Proof: The proof is rather straightforward when assuming that, for reasonable time-lengths of the surveillance epoch, the VNF profile evolution process is ergodic and stationary. This directly implies that $\bar{\gamma}_{i,(z)} = \frac{1}{K}\sum_{k=0}^{K} X[k] = \frac{1}{T}\sum_{t=0}^{T} p_{i,(z)}(t)$, where k are multiple instances of the same process whereas t are different times. This proves the lemma. □

This lemma helps to significantly reduce the number of decisional points, wherein our VNF affinity binding is executed. In particular, we can collect several profile values of the same VNF (experienced at different sample times) or different VNF instances of the same type to properly characterize a specific VNF profile. Therefore, we use all samples collected within 2 surveillance epochs in case of an alert message triggered. When gathered samples are not enough, we rely on the load forecasting process to predict future VNF profile behaviors.

F. Optimal Placement

Once the VNF affinity binding has been successfully completed, the CMS will automatically place VNFs into available compute nodes based on their profile values and their affinity group associations. This being one of main findings of our paper: the objective of our solution is to find an optimal placement that i) takes into account the statistical variance of the VNF profile values $\boldsymbol{p_i}$, ii) places the VNF in order to avoid further LCM operations, such as migrations, iii) equally balances compute nodes load to keep the system stable and to reduce the number of monitoring messages (sample points), i.e., to limit the overhead of the monitoring procedure.

We first apply the VNF placement process to VNF affinity group instances, i.e., considering the center of gravity of each VNF affinity group as a single VNF profile instance. We can formulate the following integer linear programming (ILP) problem

Problem `Proactive-VNF-Placement`:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{l \in \mathcal{L}} \log \sum_{n \in \mathcal{N}} (||\boldsymbol{c_n}|| y_{l,n}) \\
\text{subject to} \quad & \sum_{n \in \mathcal{N}} \boldsymbol{c_n} y_{l,n} \leq \boldsymbol{P_l}, \quad \forall l \in \mathcal{L}; \\
& \sum_{l \in \mathcal{L}} y_{l,n} \leq 1, \quad \forall n \in \mathcal{N}; \\
& y_{l,n} \in \{0,1\},
\end{aligned}
$$

where $||\cdot||$ is the L-1 Norm of a vector, $l \in \mathcal{L}$ is an available compute node in our system, $\boldsymbol{P_l} = \{P_{l,(z)}\}$ is the set of maximum resource availability for compute node l in terms of KPI (z) whereas $y_{l,n}$ is the binary value indicating whether the VNF class n is placed into compute node l. The log operator is needed to provide fairness between different compute node loads. While Problem `Proactive-VNF-Placement` is

Algorithm 2 Affinity-aided VNF Scheduling (AaVS)

1) Initialise $v_i = \max_{t \in \omega}(||\boldsymbol{p_i}^{(t)} - \boldsymbol{c_n}||_2 : x_{i,n} = 1, \forall i \in \mathcal{I}$.
2) Initialise set $\mathcal{H}_l = \emptyset, \forall l \in \mathcal{L}$, $\mathcal{B}_n = \emptyset, \forall n \in \mathcal{N}$, $\mathcal{F} = \emptyset$ and $l = 0$.
3) Place $i \rightarrow \mathcal{B}_n, \forall n \in \mathcal{N}$ if $x_{i,n} = 1$.
4) Sort $\mathcal{B}_n, \forall n \in \mathcal{N}$ according to v_i in a decreasing order.
5) For every n, take the first i from $\mathcal{B}_n$ and Place $i \rightarrow \mathcal{F}$ if $y_{l,n} = 1$. If i does not fit, Take the next i in $\mathcal{B}_n$.
6) Remove all i placed in $\mathcal{F}$ from $\mathcal{B}_n$. Update $\mathcal{H}_l \leftarrow \mathcal{F}$.
7) If there is any $n : \mathcal{B}_n \neq \emptyset$ then Increase $l = l+1$, Update $\mathcal{F} = \emptyset$ and go to (5).

proved to be NP-Hard [2], the solution can still be found within an affordable time as the number of variables, i.e., the number of VNF affinity groups N, is very limited. In our simulation campaign, we adopt a commercial tool, namely IBM ILOG CPLEX [24], to solve the optimization problem.

The solution optimality of Problem `Proactive-VNF-Placement` can be guaranteed if each VNF profile accurately follows the center of gravity of its assigned affinity group. In other words, the solution optimally works if the bias (variance) from the mean value of the affinity group is very low. Conversely, as soon as the VNF profile values move away from the average properties of its group the scheduling solution might fail leading to unstable system states and service disruptions. Therefore, we devise a VNF scheduling algorithm taking into account the general scheduling information of the VNF affinity groups but applying the current KPIs information of each VNF to correctly balance the compute nodes load.

The pseudo-code of our algorithm, namely Affinity-aided VNF Scheduling (AaVS), is listed in Alg. 2. Our idea is to rely on the First Fit Decreasing (FFD) algorithm [25], suggested for bin backing problems. In particular, we calculate (Line 1) the VNF profile variance as $v_i = \max_{t \in \omega} \left(||\boldsymbol{p_i}^{(t)} - \boldsymbol{c_n}||_2\right)$ along the last (at least 2) epochs ω. This is further supported by Lemma 2. Based on variance, each VNF profile value is sorted within its affinity group (Line 4), leaving at the first position the VNF profile which has experienced much variations (and might be considered as unstable). The rationale behind is that we need to first place the VNF profile which might cause (in the worst case) unexpected compute node resource outages. Iteratively, we try to schedule the other VNF profiles based on Problem `Proactive-VNF-Placement` (Line 5), i.e., based on $y_{l,n}$. Upon all VNFs have been scheduled into compute nodes l, our algorithm ends.

Assuming that the compute nodes deployment is over-provisioning, Problem `Proactive-VNF-Placement` can reasonably purse at balancing the load of compute nodes and keep them in a stable state without dangerously approaching the saturation point. Nonetheless, to avoid unexpected compute node resources saturation, $\boldsymbol{P_l}$ in Problem `Proactive-VNF-Placement` can be properly chosen by the infrastructure provider. When AaVS is applied, the fairness among different compute nodes can significantly degrade because of unpredictable VNF profile spikes. Therefore, we design a controller in charge of promptly changing the number of VNF affinity groups (and re-grouping VNFs profiles) when VNF profiles significantly differ from the VNF affinity group properties, as explained in Section III-D and empirically shown in Section IV. However, the entire process could be affected by the length of the surveillance epoch ω, which is dynamically adjusted, as explained in the next section.

G. Surveillance epoch adjustment

The decisional points play a key-role because: i) at those times the system might re-build the affinity groups and improve the accuracy of the VNF placement that, in turns, translates into a better Quality-of-Decisions and less LCM operations in the near future due to a stable system conditions, ii) complexity and overhead of the DMD are strictly related to the frequency of the decisional points, i.e., surveillance epoch length ω. An optimal trade-off must be found based on the current system conditions as well as previous observations. We design an adaptive scheme to keep track of previous alert triggers while increasing the surveillance epoch when the stability of the system can be preserved for a longer time period.

Our scheme is based on a machine-learning algorithm, namely Q-Learning [26]. The main idea behind is to learn from previous actions and obtained rewards in order to take the optimal decision in the future while pursuing the reward maximization. Without loss of generality, we define the index of surveillance epoch as well as the decisional point at the end of a surveillance epoch by $\tau \in \mathcal{T}$. Let us define the state space $\pi \in \Pi$ as the number of VNF profile deviations j experienced at the previous decisional point, i.e., $\pi^\tau = j_{(\tau-1)}$. At every decisional point τ, our system may take different actions a^τ on how much to increase (decrease) the next surveillance epoch $\omega^{(\tau+1)}$, i.e., $a^\tau = \{+k \cdot o\}$, where o is defined as the least step size. After taking an action a^τ, the system will be rewarded based on a reward function $R(\pi^\tau, a^\tau) = \frac{\omega^{(\tau)}}{j_\tau^\beta}$, where $\omega^{(\tau)}$ is the length of the surveillance epoch between two decisional points $\tau - 1$ and τ. The objective is to maximize the surveillance epoch ω while keeping low (or zero) the number of VNF profile deviations occurred in the last surveillance epoch, which might compromise the stability of our system. $\beta \leq 1$ is a tunable parameter that can be adjusted by the infrastructure provider to have a slower (faster) changing of the surveillance epoch at expense of less (more) scheduling optimality.

Our solution builds a Q-table collecting the reward coming from each possible pair (π, a) based on the following equation

$$Q(\pi, a) = (1-\alpha)Q(\pi, a) + \alpha \left[R(\pi^\tau, a^\tau, \pi^{\tau+1}) + \psi q_{\max}\right], \quad (1)$$

where $q_{\max} = \max_{a^{\tau+1}} Q(\pi^{\tau+1}, a^{\tau+1})$, and $R(\pi^\tau, a^\tau, \pi^{\tau+1})$ is the reward obtained from action a^τ leading to state $\pi^{\tau+1}$. α and ψ are the learning rate and the discount rate, correspondingly. The former balances the stored information (in the Q-table) against the current observed ones. It is usually set differently per state and evolving over time, i.e., $\alpha^\tau_{\pi,a} = \frac{0.5}{i(\pi,a)}$, where $i(\pi, a)$ is the number of times we have explored state π by time τ. The latter gives a less weight to old information, which could become incorrect. This is useful when the stationary and ergodic assumption on the VNF statistical properties could not be taken for very long periods (please refer to Section III-E). This is commonly fixed to 0.9 ([26]). When a new action must

[2] Due to the pages limitation, we skip the formal proof as the problem can be reduced in a polynomial time into a bin packing problem, known to be NP-Hard. However, we refer the reader to [23] for further details.

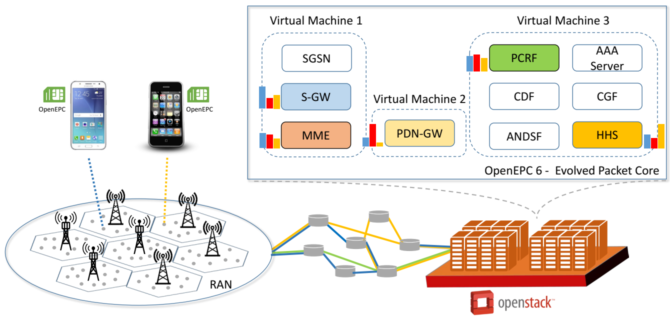


Fig. 7. Real Evaluation Case: OpenEPC mobile core.

be taken, our system may select it randomly (with probability $\phi \leq 1$) among available actions $a \in \mathcal{A}$ or it can select the one maximizing the reward (with probability $1-\phi$) based on the information stored in the Q-table, i.e., $a = \arg\max_{a^\tau} Q(\pi, a)$.

IV. Performance Evaluation

We conduct an exhaustive simulation campaign by means of a mathematical tool, such as MATLAB. All building blocks of our solution are implemented and executed using several random seeds to keep the confidence degree of our results below 0.1%. To validate our results, we evaluate a realistic use case using virtual functions deployed in our testbed. This provides a set of reference points for our VNF profiles creation process.

A. Evaluation case: OpenEPC

We implement a real network deployment with 2 NEC eNBs [27] and a virtualized core domain using a commercial software, OpenEPC [28]. Our testbed deployment is shown in Fig. 7. A number of mobile devices provided with a customized SIM-card are connected to the mobile core domain, running on OpenStack. Different KPIs for any specific VNF, such as MME, S-GW and P-GW are collected by means of Ceilometer, a telemetry software provided with OpenStack. The evaluation time window is set to 2 hours and two different user profiles are considered: *high-demanding* in case of data traffic upload and download, *low-demanding* in case of high-mobility (several hand-overs) but no data traffic (only control signal).

Overall results are summarized in Table I. We have classified only the most significant KPIs in percentage, based on the total capacity of compute nodes. Interestingly, they suggest a specific set of requirements that are exploiting throughout our performance evaluation section, ranging from low demanding requirements, e.g., HSS for *low* configuration, up to high-demanding requirements, e.g., PDN-GW for *high* configuration.

TABLE I
Virtualized Network Functions KPIs (OpenEPC 6)

VNFs	**CPU** (μ) [%]		**Mem** (m) [%]		**Net** (η) [%]	
	Low	High	Low	High	Low	High
MME	17.7	2.9	15.9	3.8	5.8	1.9
S-GW	0.7	79.1	0.3	3.3	0.14	91.2
HSS	0.9	2.9	1.1	4.5	0.7	1.3
PCRF	1.2	1.9	0.6	3.9	0.5	0.9
PDN-GW	1.7	53.1	2.1	37.2	0.8	92

B. Simulation setup

NFV system service orchestration is performed for a huge number of VNF instances. Given the unavailability of such a complex real testbed, we assess the performance of our approach by means of simulations taking into account as baseline the real values offered by OpenEPC VNFs and, at the same time, shedding the light on the impact of a large number of VNFs deployed on different compute nodes.

VNFs profiles are built based on a Pareto random distribution using the values listed in Table I. The long-tail effect of the random distribution is handled with a cap to limit VNF resource utilization to 100%. Once VNF baseline profiles are defined, every simulation time t a VNF instance is executed and VNF profile KPIs are generated and collected based on a normal distribution with mean equal to the VNF baseline profile and variance σ based on the considered scenario. If not differently stated, the simulation parameters used are listed in Table II.

TABLE II
Simulation Parameters

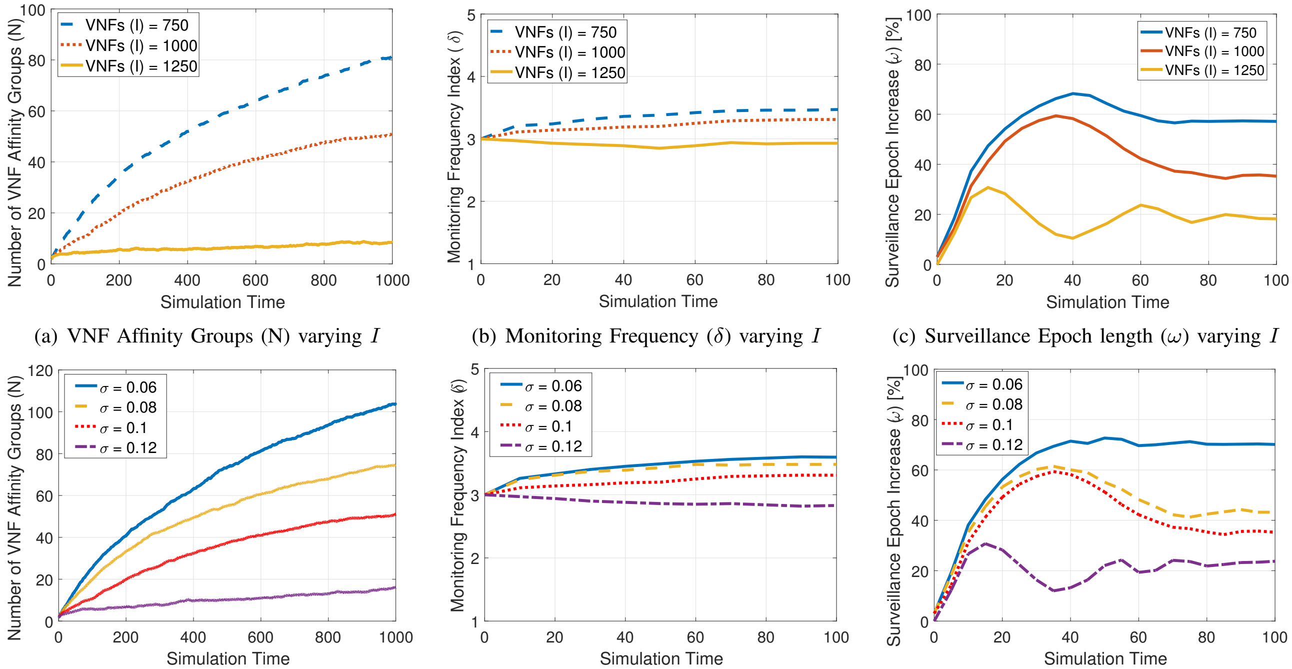
Parameters	**Values**	**Parameters**	**Values**
VNF Profiles (I)	1000	VNF Baseline Profiles	5
VNF Profile KPIs (Z)	3	VNF Profile variance (σ)	0.1
Surveillance Epoch (ω)	500t	Monitoring Interval ($1/\delta$)	$[2, 5, 10, 20, 50]$
Q-learning (β)	0.5	Simulation time	10^7t
Q-learning (ψ)	0.9	Q-learning (ϕ)	0.5

C. System parameters evolution

We study and discuss the evolution of the system parameters as well as their consistent effects on the overall system efficiency from two different perspective: i) the VNF placement and Quality-of-Decisions and ii) the reduced VNF monitoring load.

The main finding of our simulation campaign lies on the concept of VNF profile variability. VNF profile variability plays a key-role in the VNF placement and then in the overall Quality-of-Decision of the CMS. VNF profiles exhibiting significant profile deviations may result in relevant performance degradations, as the system must detect unexpected behaviours and promptly react. We study the evolution through three different adaptive parameters: the number of VNF affinity groups N, the monitoring frequency δ and the surveillance epoch length ω, as shown in Fig. 8.

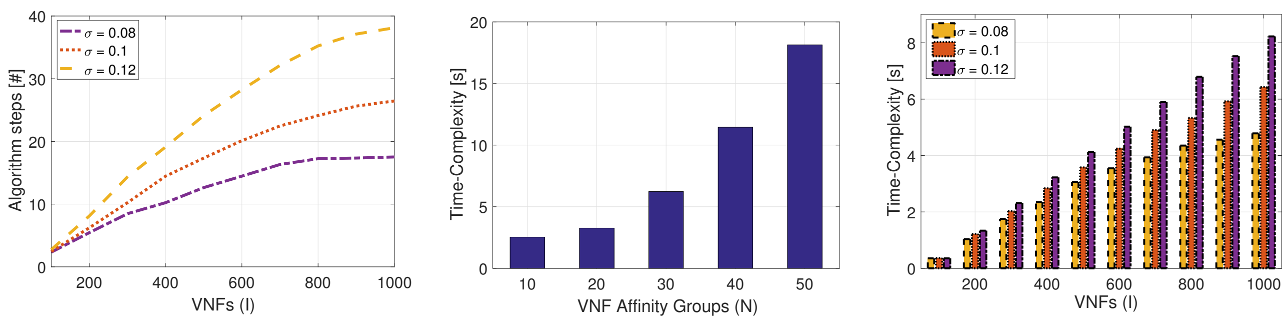
The number of affinity groups N could unveil interesting aspects. z-TORCH automatically tailors the affinity groups to specific VNF profile properties, as no VNF profile deviations occur. In other words, as soon as the unsupervised binding affinity process successfully identifies the VNF affinity groups, the granularity of such a process will be reduced in the next decisional slot to increase the accuracy of the binding. Conversely, when a failure in the binding process is detected (due to unexpected changes), the granularity of the VNF affinity groups is enlarged leading to a fewer number of groups. This is further supported by Figs. 8(a) and 8(d). On average the number of affinity groups increases quickly over time when the VNF profile variance σ is low or when a few VNF profiles are considered. When the variability becomes consistent (or the number of VNF profiles is high), the binding failures might affect the accuracy of the process keeping stable (and low) the number of VNF affinity groups.



(a) VNF Affinity Groups (N) varying I (b) Monitoring Frequency (δ) varying I (c) Surveillance Epoch length (ω) varying I

(d) VNF Affinity Groups (N) increasing σ (e) Monitoring Frequency (δ) increasing σ (f) Surveillance Epoch length (ω) increasing σ

Fig. 8. Evaluation of adaptation parameters.



(a) VNF Affinity binding using ekm (b) Proactive VNF Placement (c) VNF placement using AaVS

Fig. 9. Solution complexity analysis.

Another important feature of z-TORCH is the monitoring load which is directly triggered by the monitoring frequency δ. In our simulations, we consider a fixed set of 5 frequency intervals, i.e., the largest index 5 results in a very low monitoring load. Figs. 8(b) and 8(e) show the evolution of the monitoring index. When the statistical variance σ or the number of VNF instances is low, the system reduces the monitoring burden on average. This is due to a more stable system state. On the other side, when the number of VNF profile deviations grows, the monitoring load needs to be promptly adapted incurring in more monitoring messages.

The last parameter is the surveillance epoch length ω, which has a two-fold aspect: i) it might significantly boil down the complexity of our solution by delaying the next decisional time for making LCM decisions and ii) it impacts on the number of monitoring information accounted for the next binding affinity group operation. This parameter is driven by the Q-learning approach, as explained in Section III-G. In Figs 8(c) and 8(f), we show the effect of the VNF profile variability σ and the number of considered VNF profile instances I on the surveillance epoch length ω. High VNF profile variance values and huge number of VNF profile instances result in more unstable behaviours. When the variability of the VNF instances is limited, the surveillance epoch length on average increases (and in turn reduces the complexity of the decisional mechanisms) and stabilizes.

D. Complexity and time performance

While the adaptiveness of z-TORCH allows to promptly react to unexpected changes in the VNF profiles and to reduce the monitoring load, here we show the cost in terms of complexity of our novel mechanism. We evaluate three different novel algorithms, described in the paper. In Fig. 9(a), we show the number of steps of enhanced k-means (ekm) algorithm needed to converge. Notably, the variability of the VNF profiles might affect the complexity of the algorithm. However, the curves exhibit a sub-linear dependency on the number of VNF profile instances, which makes our algorithm suitable even for crowded VNF environment.

We next analyze the time complexity in terms of seconds for the Proactive VNF Placement problem solution using a commercial solver, namely IBM ILOG CPLEX. Specifically, we run our algorithm on a dual Intel(R) Xeon CPU 2.40GHz 4-cores and 16GB RAM. Fig. 9(b) shows the time complexity when considering a different number of VNF affinity groups. As expected the complexity of such solution grows exponentially with respect to the number of affinity groups (centres of gravity) due to the NP-Hardness property of the optimization problem described in Section III-F. However, in realistic environments the number of VNF affinity groups is low when compared to the number of VNF profile instances, which makes still the approach valid and reasonable.

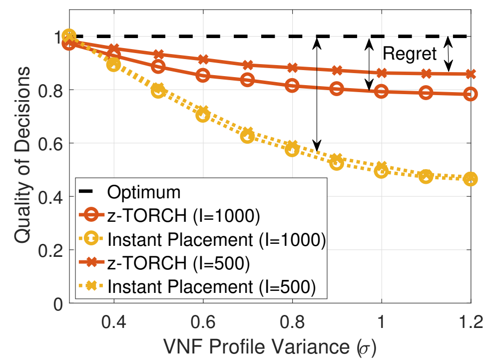
Last, we show the time complexity performance of the VNF placement algorithm, namely AaVS, as described in Section III-F. In Fig. 9(c) we depict complexity results when applied different VNF profile variance σ and an increasing number of VNF profile instances I. Interestingly, high values of σ increas the growing rate of the complexity but still showing a sublinear behavior, which in our test never exceeds 9 seconds of running time. We can conclude that AaVS can be easily applied for realistic scenarios where the number of VNF instances may dramatically grow.

E. z-TORCH: advantages and limitations

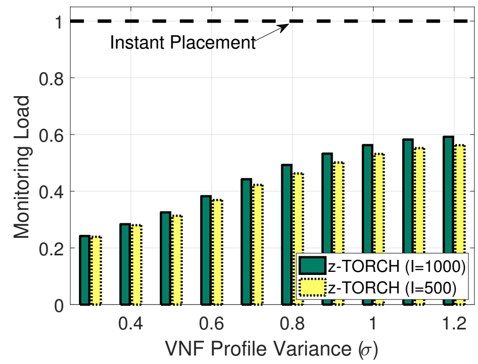
Due to the lack of existing solutions addressing jointly both optimal placement (quality of decision) and monitoring load minimization, we compare the performance of z-TORCH against a legacy approach, wherein optimal VNF placement decisions are taken every decisional time without exploiting machine-learning solutions. We call this benchmark as *Instant Placement*. Additionally, to evaluate the goodness of our solution, we have developed an optimal VNF placement solution, namely *Optimum*. This solution possesses a God-knowledge of the future VNF profile deviations. Therefore, it can calculate the optimal VNF placement (for each decisional time) in order to minimize the overall VNF migrations in the future. We denote the performance difference between our approach and the optimal one as *Regret*, following the online decisional algorithms terminology.

We evaluate our approach in terms of Quality-of-Decisions (QoD) assuming that the *Optimum* policy takes the best decision, i.e., QoD $= 1$. We use then the number of migrations performed by the optimal policy as benchmark, and we calculate the number of VNF migrations exceeding the benchmark. Resulting QoD is the ratio between the optimal number of migrations and the number of migrations required by each solution. In Fig. 10(a), we show the QoD results while varying the VNF profile variance σ for two different scenarios with 500 and 1000 VNF profile instances. Interestingly, when the variance is very low, i.e., VNF profiles are predictable and stable, the Instant Placement solution slightly outperforms z-TORCH. This is due to the initial training phase in which z-TORCH needs to adapt and stabilize. When the VNF profile variance increases, z-TORCH shows a near-optimal results (up to 88.6%) almost doubling the performance of the Instant Placement solution.

Last, we show the monitoring load analysis when z-TORCH is in place. In this case we only compare against the Instant Placement, as the optimum solution is executed only once. Instant Placement can be considered as the worst case since it needs monitoring information every sample point (δ). Therefore, we normalize the number of monitoring messages needed by z-TORCH by the ones needed by the Instant Placement solution. Results are depicted in Fig. 10(b). The larger the variability of VNF profiles increases, the higher the monitoring load. This is due to a number of VNF profile deviations, which must be controlled through more monitoring information. However, the monitoring load seems to stabilize around $50-60\%$ even for significant variance values σ.



(a) Quality of Decisions



(b) Monitoring Load

Fig. 10. z-TORCH: Placement and monitoring performance.

This confirms that z-TORCH outperforms legacy solutions while showing near-optimal performance at low monitoring costs. Nonetheless, considered solutions (Instant Placement and Optimum) requires a huge complexity making them not suitable for being executed in an affordable time.

V. Deployment Considerations

In this section we will provide insights at various deployment and implementation considerations with respect to standard ETSI NFV MANO system [5], which is a standard CMS for NFV based environment. The NFV MANO system is composed of three main functional bocks namely the Virtualized Infrastructure Manager (VIM), VNF Manager (VNFM) and NFV Orchestrator (NFVO). The ETSI NFV MANO system is designed to manage and orchestrate virtualized resources in an NFV Infrastructure (NFVI) such as virtualized compute, network, storage, memory etc via the VIM. It also manages the individual VNFs that are deployed over the NFVI via the VNFM. The NFVO is designed to perform resource orchestration and service orchestration, where the service meant here is the Network Service (NS) that is formed by the concatenation of multiple relevant VNFs to

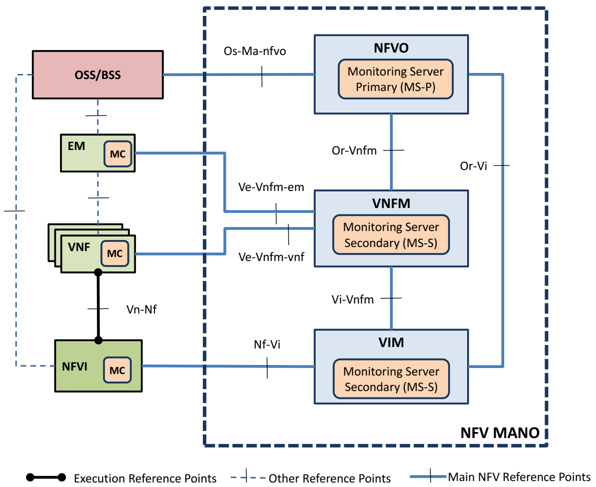


Fig. 11. NFV MANO system with integrated monitoring system (Distributed).

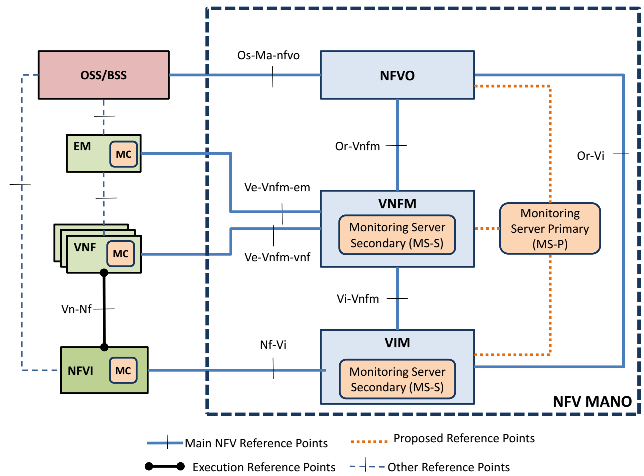


Fig. 12. NFV MANO system with integrated monitoring system (Centralized).

provide a composite network service. In other words the VIM, VNFM and NFVO constitute the CMS. There are no specific proposals as to how the monitoring system will be integrated in the NFV MANO system. It is implied that the VIM, VNFM and the NFVO will monitor their respective layers for performance and fault management and take relevant LCM decisions as per the logic local to the respective functional block. There is also a requirement to monitor the MANO functional blocks for its own performance/fault management and that there is indeed a requirement to have a monitoring entity i.e., MANO Monitor, with which all the three MANO functional elements will interact with [29]. However, there is no specific architectural proposal. In view of the prevailing understanding, there are thus two layers of monitoring for performance/fault management; Layer1 is for the monitoring of the virtualized infrastructure and resources, while Layer2 is for the monitoring of the MANO functional blocks themselves. In this regard we propose two possible deployment options for integrating a monitoring system within the ETSI NFV MANO framework that can then be leveraged by the proposed z-TORCH method.

A. Deployment Option 1

This option is illustrated in Figure 11, where the MS is integrated within each MANO functional blocks while the MCs are deployed within the virtualized infrastructure/resources. As explained above, the MC will be configurable by the MS. The key difference is that due to the distribution of the MS inside the MANO functional blocks, the MS within the NFVO is the Primary MS (MSP), while the MS inside the VIM and VNFM are the Secondary MS (MSS). The MSSs can independently monitor, collect, analyze data from the functional block respective layer. For example, the MSS within the VNFM will be able to deploy and configure MC instances inside the VNF instance(s) and will also independently collect and locally analyze monitored data from these MCs. Based on the analysis of the monitored data, the VNFM can take VNF specific LCM decisions as per the policy/decision logic local to the VNFM. Similarly the MS-S inside the VIM will deploy and configure MC within the virtualized/non-virtualized infrastructure resources (e.g., compute, network, memory, storage) and monitor and manage them as per its local policy/decision logic. However, the LCM decisions taken by the VIM and/or VNFM must be validated by the NFVO as the latter has an overview of the overall NS that is composed of several VNFs managed by possibly different VNFMs and deployed over possibly different VIM platforms.

Owing to the level and centrality of the NFVO in the LCM decision process; the MS-P is integrated within NFVO. The MS-P does not deploy/configure/monitor any specific MCs but it monitors and configures the MS-S instances in VNFM and VIM. The MS-P may override any configuration parameter within the MS-S instances at any time. Our proposed method shall typically run inside MSP and based on the feedback it receives from MS-S will (re)compute and (re)adjust the values of ω and/or δ and/or t for the specific MSS instances. Based on these values, the MSS will (re)configure the MC instances within their respective monitoring domain. The MSP will also configure the MS-S with the KPIs to monitor and can change the configuration parameters of the MS-S any time. The MS-P, based on the inputs received from the MSS will forward them to the analysis engine (AE). The AE after analyzing the data send the results to the decision engine which will take appropriate decision on LCM, recompute the necessary configuration parameters for the MSS instances and push them over the respective standard reference points i.e., OrVi and OrVNFM reference points. Please note that the AE and DE components and their inter-relationship with themselves and the MSP is similar to what is shown in Figure 2.Our proposed method can either run in the MS-P or the AE and the AE then provide the recommended configurations parameters to the MS-S.

B. Deployment Option 2

This option is depicted in Figure 12. In this deployment, the MS-P is a central entity that interacts with the MS-S located in VNFM and VIM functional blocks. The NFVO does not carry a MS-S as it will interact with the external MS-P. The method described here is implemented in MS-P that will then be used to compute the relevant configuration parameters for the MS-C, which in turn will configure the MCs of their respective domains. In this case the NFVO, which carries the

AE and the DE (see Figure 2) will inform the MS-P of its LCM decision and also the identities of the VNFs and NSs that has been affected, and based on this information the MS-P will (re)calculate the relevant configuration parameters and push then to the MS-Ss so that they can configure the MCs within their respective layer. It is also possible that the MS-P may derive separate configuration values for the MS-S. This will make the MC at the NFVI and VNF level to use different monitoring configuration.

VI. Conclusion

In this work we have designed an automated solution, namely z-TORCH, performing joint NFV orchestration and monitoring re-configuration operations without requiring human intervention. We have built our solution based on machine-learning approaches. In particular, we have proposed an unsupervised binding affinity solution to study and properly profile VNF KPIs. This has allowed us to pro-actively place VNFs into compute nodes pursuing the quality-of-decision (QoD) maximization and, in turn, the decisional complexity minimization. In addition, with z-TORCH we automatically adapt the VNF monitoring load according to VNF profile time variations.

The main characteristics of our proposed z-TORCH solutions can be summarized as follows: i) an unsupervised system in charge of profiling VNF KPIs based on previous monitoring information, ii) a proactive VNF placement based on pre-calculated affinity groups, iii) an adaptive monitoring load control to minimize the overhead of monitoring information. NP-Hardness proofs and heuristics algorithms are introduced to make our framework practical and implementable. An exhaustive simulation campaign has been carried out to validate our solution against a legacy system showing that z-TORCH can achieve near-optimal results at very limited monitoring costs.

References

[1] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, and M. Wolf, "Monalytics: Online monitoring and analytics for managing large scale data centers," in *Proceedings of the 7th International Conference on Autonomic Computing*, ser. ICAC '10. ACM, 2010, pp. 141–150.

[2] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch, "Cost analysis of initial deployment strategies for virtualized mobile core network functions," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 60–66, Dec 2015.

[3] F. Z. Yousaf, C. Goncalves, and L. Moreira-Matias, "RAVA Resource aware VNF agnostic NFV orchestration method for virtualized networks," in *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2016.

[4] ETSI NFV ISG, "GR NFV-IFA 022 V0.7.1: Network Function Virtualisation (NFV); Management and Orchestration;Report on Management and Connectivity for Multi-Site Services," 2017.

[5] ——, "GS NFV-MAN 001 V1.1.1 Network Function Virtualisation (NFV); Management and Orchestration," Dec. 2014.

[6] H. Huang and L. Wang, "P&P: a combined push-pull model for resource monitoring in cloud computing environment," in *2010 IEEE 3rd International Conference on Cloud Computing*, July, pp. 260–267.

[7] Y.-A. L. Borgne, S. Santini, and G. Bontempi, "Adaptive model selection for time series prediction in wireless sensor networks," *Signal Processing*, vol. 87, no. 12, pp. 3010–3020, 2007.

[8] H. Malik, A. S. Malik, and C. K. Roy, "A methodology to optimize query in wireless sensor networks using historical data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2, pp. 227–238, 2011.

[9] G. M. Dias, T. Adame, B. Bellalta, and S. Oechsner, "A self-managed architecture for sensor networks based on real time data analysis," in *2016 Future Technologies Conference (FTC)*, Dec 2016, pp. 1297–1299.

[10] G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting sampling interval of sensor networks using on-line reinforcement learning," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 460–465.

[11] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *2015 IEEE International Conference on Communications (ICC)*, 2015.

[12] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec, pp. 1–6.

[13] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 255–260.

[14] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for load balancing in NFV networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017.

[15] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise wlans," in *IFIP/IEEE International Symposium on Integrated Network Management*, May 2015, pp. 1220–1225.

[16] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, "An architecture for on-demand service deployment over a telco cdn," in *2016 IEEE International Conference on Communications (ICC)*, May, pp. 1–6.

[17] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5G carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, March 2016.

[18] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*.

[19] "Zabbix - The Enterprise Class Monitoring Platform," http://www.zabbix.com, 2017.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[21] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract)," in *Proceedings of the Tenth Annual Symposium on Computational Geometry*, ser. SCG '94. ACM, pp. 332–339.

[22] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, ser. SCG '06. ACM, 2006, pp. 144–153.

[23] H. Kellerer and U. Pferschy, "A new fully polynomial time approximation scheme for the knapsack problem," *Journal of Combinatorial Optimization*, vol. 3, pp. 59–71, 1999.

[24] IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/us-en/marketplace/ibm-ilog-cplex/.

[25] G. Dósa, R. Li, X. Han, and Z. Tuza, "Tight absolute bound for first fit decreasing bin-packing: FFD(L)$\leq$11/9opt(l)+6/9," *Theor. Comput. Sci.*, vol. 510, pp. 13–61, Oct. 2013.

[26] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, 1992, pp. 279–292.

[27] NEC LTE small-cell MB4420. http://www.nec.com/en/global/solutions/nsp/sc2/prod/e-nodeb.html.

[28] OpenEPC 6. http://www.openepc.com/.

[29] ETSI NFV ISG, "GR NFV-IFA 021 V0.9.0: Network Function Virtualisation (NFV); Management and Orchestration;Report on management of NFV-MANO and automated deployment of EM and other OSS functions," 2017.