OPEN ACCESS

University of BRISTOL

Bunyakitanon, M., Vasilakos, X., Nejabati, R., & Simeonidou, D. (2020). End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning. *IEEE Transactions on Cognitive Communications and Networking*, *6*(2), 534-547. https://doi.org/10.1109/TCCN.2020.2988486

Publisher's PDF, also known as Version of record

License (if available):
CC BY

Link to published version (if available):
10.1109/TCCN.2020.2988486

Link to publication record in Explore Bristol Research
PDF-document

This is the final published version of the article (version of record). It first appeared online via Institute of Electrical and Electronics Engineers at https://ieeexplore.ieee.org/document/9070195/keywords#keywords . Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/

End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning

Monchai Bunyakitanon, Xenofon Vasilakos, Reza Nejabati, *Senior Member, IEEE*, and Dimitra Simeonidou, *Fellow, IEEE*

***Abstract*—The autonomous placement of Virtual Network Functions (VNFs) is a key aspect of Zero-touch network and Service Management (ZSM) in Fifth Generation (5G) networking. Therefore, current orchestration frameworks need to be enhanced, accordingly. To address this need, this work presents an Adapted REinforcement Learning VNF Performance Prediction module for Autonomous VNF Placement, namely AREL3P. Our solution design bears a dual novelty. First, it leverages end-to-end service-level performance predictions for placing VNFs. Second, whereas the majority of other Machine Learning efforts in the literature use Supervised Learning (SL) techniques, AREL3P is based on a particular form of Reinforcement Learning adapted to predictions. This makes placement decisions more resilient to dynamic conditions, as well as portable to other network nodes, and able to generalize in heterogeneous network environments. Backed by a meticulous performance evaluation over a real 5G end-to-end testbed, we verify the above properties after integrating AREL3P to Open Source Management and Orchestration (OSM MANO) decisions. Among other highlights, we show increased VNF performance predictions accuracy by 40–45%, and an overall improved VNF placement efficiency against other SL benchmarks reflected by near-optimal decision scores in 23 out of a total of 27 investigated scenarios.**

***Index Terms*—Machine learning, network function virtualization, end-to-end communication, zero-touch management, cloud and edge computing.**

I. Introduction

CONTEMPORARY networks are becoming increasingly programmable based on two key concepts: (i) Software-Defined Networking (SDN) and (ii) Network Function Virtualization (NFV). SDN facilitates network management and network configuration by enabling networks to be directly programmable, whereas NFV decouples Network Functions (NFs) from special-purpose dedicated hardware by virtualizing them into software building blocks. Building on top of SDN and NFV, fifth generation networks (5G) and, in particular, their network and service management, are envisioned to be fully autonomous based on the Zero-touch network and Service Management (ZSM) concept, as the next-generation management and operation model defined by ETSI [1]. At the same time, recent advances in Machine Learning (ML) have enhanced the performance of corresponding ML techniques as well as their applicability in real-life problems, thus enabling intelligent and autonomously operated systems such as in the case of autonomous vehicles.

As we approach closer to the 5G era, the combination of network programmability and enhanced ML models opens a promising and wide spectrum of ML in Networking (MLN) applications against emerging issues like ZSM and more traditional –yet challenging– ones such as traffic prediction, dynamic traffic steering, and dynamic resource management after Quality-of-Service (QoS) or Quality-of-Experience (QoE) requirements.

Currently proposed MLN applications face two big concerns, namely, (i) the *lack of model adaptability* and the related (ii) *(in)feasibility* to apply in practice due to the high costs implied for training and maintain the models. To understand these concerns, we need to focus on two aspects. First off, large 5G networks such as the 5GinFIRE[1] and 5GCity[2] testbeds are usually custom-made in order to achieve enhanced performance levels. Due to being custom-made and configured, the corresponding Supervised Learning (SL) models can not generalize sufficiently and their performance is significantly degraded when applied to other networks or even to other parts of the same network.[3] Second, the inborn dynamicity and natural evolution of all networks (particularly, 5G) forces SL models to become quickly outdated even for the networks that they were trained for. Due to this lack of adaptability in both points above, many models need to be not only specially trained, but also specially maintained, in some cases even per deployed model $\mathrm{instance}$[3]. This implies a very high cost, challenging the feasibility of applying such solutions in practice.

The above concerns raise the following important question: "How can we train MLN models that can be (i) adoptable (i.e., able to generalize and to tolerate network dynamics), (ii) feasible (i.e., cost-efficient w.r.t. training and

Manuscript received August 30, 2019; revised December 11, 2019 and April 5, 2020; accepted April 8, 2020. Date of publication April 17, 2020; date of current version June 9, 2020. This work is funded by the EPSRC Grant EP/L020009/1: TOwards Ultimate Convergence of All Networks (TOUCAN). The associate editor coordinating the review of this article and approving it for publication was C. X. Wang. *(Corresponding author: Monchai Bunyakitanon.)*

The authors are with the Smart Internet Lab, University of Bristol, Bristol BS8 1UB, U.K. (e-mail: m.bunyakitanon@bristol.ac.uk; xenofon.vasilakos@bristol.ac.uk; reza.nejabati@bristol.ac.uk; dimitra.simeonidou@bristol.ac.uk).

Digital Object Identifier 10.1109/TCCN.2020.2988486

[1]https://5ginfire.eu

[2]https://www.5gcity.eu

[3]For instance, our comparative evaluation study in the context of our Smart City Safety (SCS) use case shows that SL-based benchmark models fail completely to adopt to other VNF hosting nodes of the same network. This means that they need to be not only trained per node, but also maintained per node to capture local network conditions and dynamics.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

maintaining the models), and (iii) sufficiently accurate for the purposes of ZSM in 5G?" Motivated by this question, the current work proposes an Adapted REinforcement Learning VNF Performance Prediction module for Autonomous VNF Placement (AREL3P). The purpose of this module is to enhance network orchestrator systems based on online learning. As we discuss in Section II, both the problem of autonomous Virtual Network Function (VNF) placement and the adopted approach of Q-learning are contemporary, interesting and challenging. Most existing approaches are SL-based and focus mainly on local system-level monitoring information and performance predictions. Unlike that, ours sees the "greater view" by providing accurate end-to-end (e2e) service-level performance predictions to orchestrators like the Open Source MANagement and Orchestration framework (OSM MANO)[4] or Openstack.[5] To the best of our knowledge, well-established orchestrators possess solid VNF placement mechanisms, yet most of them without any native support for ML modules including Reinforcement Learning (RL). AREL3P covers this gap by serving as an extension module to orchestrators, being embedded with its own e2e service-level monitoring and corresponding intelligent VNF performance prediction mechanism at candidate VNF hosting nodes.

A. Contribution

The main contributions of this article can be summarized as:

1) *A novel, adoptable, feasible and accurate RL-based approach to VNF placement:* Despite the increasing interest of the networking community on ML, only a few efforts focus on RL for autonomous VNF placement, such as in [2], [3], [4], [5], (see discussion in Section V). Most solutions use SL for resource allocation such in [6], [7]. As verified by our evaluation results, this leads to costly models that can *not* generalize without significant performance degradation, nor can keep up-to-date with network dynamics. To cover this gap, we adopted a Q-learning scheme in our novel AREL3P solution. This particular type of RL exhibits *good resilience* to network dynamics over a realistic testbed environment and use case setup. Moreover, our real-testbed evaluation results show that AREL3P can also generalize well, hence it can address adaptability concerns. Last, our approach is feasible in practice due to achieving a good trade-off between the predictions accuracy and the costs for model training and maintenance.
2) *A novel approach to autonomous VNF placement based on e2e service-level predictions:* Most works in the literature, including ML-based ones, narrow their focus on low/system-level monitoring and predictions, thus missing a holistic, e2e, service-level point of view. To cover this gap, our approach monitors and forecasts e2e service-level performance across system layers spanning from the network layer up until the application layer.
3) *Real testbed and use case-based evaluation & validation:* Unlike most past studies on VNF placement that use simulation environments, we present *real-life* experimental results with all performance evaluation and validation experiments conducted over 5GinFIRE testbed at the University of Bristol (UNIVBRIS) [8]. We use the OSM MANO and take a use case-driven approach by adopting the dynamic environment scenario defined in the SCS [9] use case, which involves an e2e application running VNF video transcoding. Note though, that the applicability of our solution extends to all e2e services beyond this use case and scenario, and to other orchestrators apart from OSM MANO.
4) *Meticulous performance evaluation study including SL-based models, and compliance to ZSM:* We engage into different scenarios and show that the integration of AREL3P to OSM MANO improves the efficiency of its VNF placement decisions while complying to the ZSM concept in terms of *self-healing*. Besides our own model, we trained and deployed five well-known SL-based models, which we studied both separately as well as in a head-to-head comparison against AREL3P. Our findings denote both a better ability to *generalize* and a *higher resilience* of AREL3P under dynamic network conditions where nodes can be added or depart at any time (a.k.a. high node churn).
   Regarding static network circumstances, the custom-trained SL-based benchmark models exhibited a higher predictions accuracy compared to AREL3P. This is largely due to the non-realistic assumption of candidate hosting nodes (physical or Virtual Machines (VMs)) and resource demand levels remaining static over time. Even more importantly, the higher accuracy achieved by the SL models did not translate to more efficient VNF placement decisions by OSM MANO compared to using AREL3P. In further, AREL3P exceeds the performance of both (i) a traditional and (ii) a random VNF placement method used also as benchmarks, due to combining resource availability and a good accuracy level of VNF performance predictions.

B. Outline

The remainder of this paper is organized as follows. Section II discusses the background and motivation. Section III describes the system model. Section IV provides a meticulous discussion of our system evaluation and validation. Section V is dedicated to the state of the art. Finally, we conclude this paper and refer to our future work goals in Section VI.

II. Background and Motivation

VNFs are in essence VMs that work as NFs. Their placement is on its own one of the most challenging problems that emerged with modern programmable networks, hence the significant amount of work in the literature (see Section V). Although the efficiency of VNF placement depends on processing power and network performance, most existing placement algorithms merely consider resource availability for

[4]https://osm.etsi.org/
[5]https://www.openstack.org/

selecting a VNF host. In practice though, this approach does not provide any performance guarantee for the VNFs for a number of reasons including the dynamic nature of network demand. Another aspect which hinders ZSM, regards the engagement of human decisions in the loop. Despite their powerful VNF placement systems, today's well-established orchestrators still call for human interaction.

This reality calls for exploring VNF placement from a different perspective: an (i) *autonomous* and *intelligent* one that (ii) learns in operation in order to adapt to network dynamics and (iii) leverages accurate service-level *end-to-end* performance predictions. This combination of intelligence and resilience to dynamics results in autonomous VNF placement in accordance to ZSM, thus increasing the level of interest and motivation for exploring the potential of an appropriate ML technique.

Networks are typically composed of a variety of servers, client nodes, switches and routers. Some of them are static, whereas a continuously increasing number of them in 5G are virtual and/or mobile. Particularly the latter two types of nodes can be added, depart or fail at any time (a.k.a. "node churn"). Not only that, but also each node possesses different hardware and software capabilities. Therefore, VNF placement is difficult to address and beyond just considering the capabilities of VNF hosting nodes. Moreover, predicting VNF performance using the same ML model for all nodes is inaccurate. Most network nodes need a ML model trained (tailored) for them to achieve a high level of prediction accuracy. Nevertheless, this is *not* cost-inefficient because data need to be sufficiently large and comprehensive to allow the algorithm to learn individually for every candidate hosting node. The problem is even harder for dynamic nodes whose status change with time due to churn or other changes happening to their connected devices. This so-called "Concept Drift" affects the prediction model over time in an unpredictable way [10].

A. Why Adopting Reinforcement Learning

Despite the increased interest of the research community on applied ML for networking, only a limited amount of effort such as in [2], [3] has been put on using RL for managing VNF resources; and even in these works, without any consideration to end-to-end service performance. This gap is largely due to the fact that applying RL in networking constitutes a non-trivial task. It carries diversity and the complexity of both the concept of RL and the networking problems it tries to address. The greatest challenges refer to the creation of the RL model itself, on how to monitor the necessary input parameters, and on how to feed these parameter values to the RL model.

Our approach adopts a particular form of RL, *Q-learning*. Q-learning is an off-policy RL algorithm that seeks to learn a placement policy that maximizes a target reward while minimizing the end-to-end service delay by being continuously updated in operation. This makes AREL3P fundamentally different than most ML-based solutions in the literature, which are SL-based, thus they are tightly coupled to their training nodes. As a result, these models can not generalize to other nodes, they raise both feasibility and adaptability concerns (as discussed in the introduction), and in general show limited or no resilience to new/dynamic network conditions.

Regarding the limited number of past work in the literature that uses RL [2], [3], these works focus on classification or on policy selection. Unlike that, our approach lies in the fact that we employ Q-learning to solve a regression problem. What is more, AREL3P can take advantage of end-to-end service-level information and other metrics, which are generally neglected by both ML and non-ML based frameworks when forecasting VNF performance on the basis of its placement. As we show in our realistic evaluation, this enhances the efficiency of placement decisions made by the OSM MANO.

Finally, our approach can work as a cross-platform prediction model for end-to-end VNF communication performance, exactly because it uses end-to-end service-level information, rather than system-level information. Note that using service-level information helps further to generalize to different hosts other than the ones the model was trained. All above mark the use of Q-learning as both *interesting* and *challenging*, not only due to its advantages over both traditional and SL-based models, but also due to applying Q-learning in networking itself.

B. Q-Learning

RL is one of the main types of ML alongside SL and Unsupervised Learning (UL) It uses online data to train a model that learns (i) to achieve a certain goal based on the cumulative positive outcome of a series of actions and (ii) to avoid mistakes stemming from negative action results. This capability allows trained RL models to be updated in operation, which can help to maintain accuracy levels despite the changes that take place in the network environment, e.g., in a network node.

Q-learning, in particular, is an off-policy RL algorithm that seeks for the best next action given the current state. This is done with the use of the *Q-learning function*, which is designed to maximize a *reward*. More specifically, the applied model explores an unknown environment by executing an action and then learning from feedback regarding the state change stored in a table, namely, the *Q-table*. Q-Table contains the states after actions and the implied action rewards for the action. Consequent actions are taken based on current state and the current knowledge, while the environment is (usually) modeled as Markov Decision Process (MDP) defined by the following five tuples:

$$M = (S, A, P_a, R_a, \gamma), \tag{1}$$

where:

- S is a finite set of states;
- A is a finite set of actions;
- P_a is the transition probability from state s at time t to state s' at time $t + 1$ after action a (see formula (2));
- R_a is the immediate reward after moving from state s to s' after action a (see formula (3));
- γ is a discount factor $0 \leq \gamma \leq 1$ that describes the importance of future rewards on the current decision.

$$P_a\big(s, s'\big) = P\big(s_{t+1} = s' | s_t = s, a_t = a\big) \tag{2}$$

$$R_a(s, s') = E(s_{r+1}|s_t = s, a_t = a, s_{t+1} = s') \quad (3)$$

The optimal decision policy for the next steps, namely $\pi(s)$, used for the selection of action a at state s is calculated after the expected return of the following value function:

$$V_\pi(s) = \max_\pi E\left(\sum_{t=0}^{\infty} \gamma^t R_t(s, s')\right) \quad (4)$$

In dynamic network environments, the entire domain is not known and keeps changing over time. Therefore, the MDP problem cannot be solved with dynamic programming methods. However, Q-learning is model-free RL and can be used to identify the optimal policy at any time t because it does not require the entire knowledge of the environment. A Q-learning agent constantly takes an action a in a state s for each time t, observes the rewards R_a and the state transitions s' and, finally, updates the Q value using the weighted average of the old and the new Q values. Formula (5) below shows how Q values get updated:

$$Q^{upt}(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_{\mathrm{a}} Q(s_{t+1}, a)\right), \quad (5)$$

where:

- $Q(s_t, a_t)$ is the old Q value and $Q^{upt}(s_t, a_t$ the updated one,
- α is the *learning rate* ranged as $0 < \alpha \leq 1$;
- r_t is a reward received from action a_t;
- γ is the discount factor defined above (see formula (1));
- $\max_{\mathrm{a}} Q(s_{t+1}, a_{t+1})$ is an estimation of the *optimal* future value, which means that $r_t + \gamma \cdot \max_{\mathrm{a}} Q(s_{t+1}, a)$ as a whole denotes the *learned value*.

Last, we note that if $\gamma < 1$, then the action values are finite even for problems that contain infinite loops.

We return and discuss how we adopt Q-learning in our solution in Section III-B including action selection strategies.

C. The Smart City Safety Use Case

As stated earlier, we take a use case-driven approach to study and exhibit the performance merits of AREL3P. Smart City Safety (SCS) [9] is defined in the context of the EU research project 5GinFIRE, which enhances public safety by providing mobility and flexibility to surveillance systems. It uses VNF and ML to enable the dynamic allocation of processing units for live stream 360^o video. SCS is latency sensitive and requires optimal VNF placement to ensure QoS.

Our corresponding system is composed of three node types, namely: a Source Node (SN), i.e., a 360 degrees camera and a Raspberry Pi (Raspi); a Processing Node (PN), i.e., a VNF video transcoder running at a server in the cloud/edge; and an End-user Node (EN), i.e., a generic computer such as laptop or tablet. All components are connected to an access point wirelessly. The workflow starts from the camera recording the video and streaming it to the Raspi. The Raspi converts the proprietary format video to standard format video allowing the video to be processed afterward and transmits it to the VNF. Then, the VNF splits this video into frames, executes face detection and face recognition, assembles these frames into a video. Finally, the processed video is transmitted and displayed at the EN.

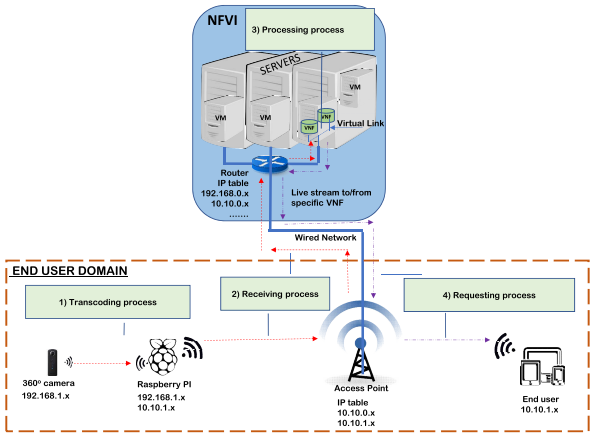

Fig. 1. Smart City Safety architecture and Workflow.

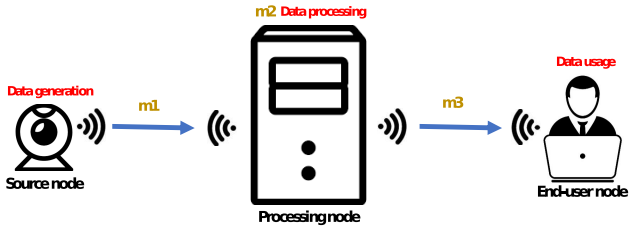

Fig. 2. Generic workflow of End-to-End services.

III. System Model

Figure 2 portrays a generic end-to-end workflow, which we adopt here for our SCS use case described earlier. As shown in the figure, a camera SN sends raw video data to some PN which in turn processes the data and transmits the output to a service end-user device such as laptop, namely, the EN.

In essence, the PN is the middle point that holds sufficient processing capabilities compared to the SN and the EN. Therefore, it is selected to place a VNF for performing, e.g., transcoding or image processing based on some policy.

We return and explain Q-learning in further in the context of our proposed adopted RL scheme in Section III-B.

Given this context, VNF performance is defined after the *total* response time of the end-to-end service T_{tot}. This embodies the overall time needed to complete every task in the end-to-end workflow of Figure 2 from the moment that SN starts to transmit raw data up until the EN has received the result of the VNF processing, i.e.,

$$T_{tot} = T_{rcv} + T_p + T_{req} \quad (6)$$

where T_{rcv} is the time spent during data generation (phase $m1$); T_{p} is the time spent during data processing

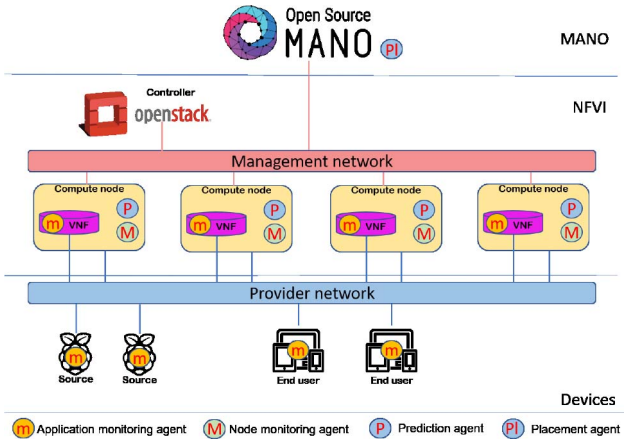

Fig. 3. High-level architecture of AREL3P.

(phase *m*2); and T_{req} is the time spent for receiving the processed results from the VNF (phase *m*3).

A. Architecture Overview

Figure 3 presents an overview of the AREL3P architecture, which comprises of three layers, namely: (i) Devices, (ii) Network Function Virtualization Infrastructure (NFVI) and (iii) OSM MANO. The devices layer lies at the low level of this architecture where end-point devices are located. The NFVI holds the role of the middle layer between Devices and OSM MANO where the cloud/edge components are installed. OSM MANO lies at the top layer as the high-level orchestration framework. Devices and NFVI are connected through the provider network. From a user's perspective, this is the main network that provides connectivity to every component of the whole end-to-end service, including devices and cloud/edge resources. In contrast, OSM MANO plays a different, crucial role from an administrator's side by interlinking the NFVI and OSM MANO layers, thus allowing management and orchestration of NFVI resources through the Virtualized Infrastructure Manager (VIM) or the OSM MANO.

In further, AREL3P applies four different agents in order to improve VNF placement at the OSM MANO: an application monitoring agent (*m*), a node monitoring agent (*M*), a prediction agent (*P*) and a placement agent (*Pl*). In more detail, agent (*m*) is collocated with the VNF to keep track of all information regarding application performance by gathering the performance values from the SNs, the PNs where the VNF and *m* itself actually reside, and the ENs. Agent (*M*), on the other hand, periodically monitors the resource utilization of a compute node and saves it for further use (see Table I). Agent *m* feeds this information to agent *P*, based on which *P* predicts the VNF performance and sends it to OSM MANO. Finally, the placement agent (*Pl*) puts the VNF to the best location.

In this work, we adapt the generic workflow of Figure 2 to the SCS use case that involves an end-to-end application running VNF video transcoding. Accordingly, we (re)define the following times per service phase:

TABLE I
NOTATIONS. INCLUDES MONITORED PARAMETERS

Notation	Description	Units	Agent
T_{trans}	video transcoding time	second(s)	m
T_{rcv}	time to receive video from Raspi	second(s)	m
T_{p}	processing time	second(s)	m
T_{req}	response to the UE time	second(s)	m
T_{tot}	total response time	second(s)	m
CPU%	CPU utilization	percentage	M
Memory available	–	gigabyte(GB)	M
Core available	–	cardinal number	M
Disk available	Available disk space	gigabyte(GB)	M

- T_{trans}*:* This is the time that is used to transform the video of the camera from proprietary format to a standard format.
- T_{rcv}*:* This is the time that a frame is transmitted from a Raspberry Pi (Raspi) node to the VNF located in the cloud/edge.
- T_p*:* This is the time that it takes for face recognition VNF to detect and recognize faces in a frame.
- T_{req}*:* This is the time need to live stream the video or the response to the User Equipment (UE) after the face processing at the VNF has ended.

The performance metrics of SCS indicate the efficiency of the system to deliver the service through above processes. Therefore, generic metrics described above have been determined to facilitate the calculations and the T_{tot} is computed by formula (7).

$$T_{\text{tot}} = T_{\text{trans}} + T_{\text{rcv}} + T_{\text{p}} + T_{\text{req}}. \tag{7}$$

B. Adapted Q-Learning

AREL3P is based on adaptive Q-learning to predict T_{tot} for assessing the expected VNF performance of an end-to-end service. The T_{tot} reflects the efficiency of the transmission between nodes and the processing power of the VNF. The data time series

$$D = \{d_1, d_2, \ldots, d_n\} \tag{8}$$

used for training AREL3P come from monitoring the service during execution time by agent *m*, where each monitoring sample d_i refers to the executed total response recorded at time *i*. Therefore, and according to the definition provided in (6), each d_i is a tuple of the three time samples: $\{(T^i_{rcv}, T^i_p, T^i_{req}))\}$. As we discussed in Section II-B, Q-learning has three key elements reflected in the Q value update formula (5), namely: State, Action and Reward.

1) State: The State space (*S*) comprises all the possible performance prediction states of T_{tot}. The current state of performance $s \in S$ provides the basis of information (real current value of T_{tot}) for each possible next performance state. The possible next states s' for predicting of the prediction are:

- *Less than:* is a state where the Predicted Value (PV) is less than the Real Value (RV).
- *Equal:* is a state where the PV is equal to the RV.
- *More than:* is a state where the PV is more than the RV.

2) Action: The actions set $A(s)$ in AREL3P contains actions a with the increment or the decrement of the prediction of

T_{tot} by a step of 0.01. According to the SCS use case, the minimum-maximum value we can predict for T_{tot} is 0-2 s.[6]

3) Reward: Rewards $R_a(s, s')$ are immediately obtained after executing an action that changes the state from s to s', reflecting the quality of the prediction in the new state. AREL3P gives immediate rewards in the range of [0, 1]. For that, we define an acceptable error margin, hence AREL3P adapts the *epsilon intensive band* (ϵ). This ϵ serves as a tolerance margin[7] constructed by boundary lines at distance ϵ from a hyperplane that separates rewards in two regions. First, a region of values in (0, 1], which is given when the prediction error falls within a distance ϵ from the RV. Note that the maximum reward, i.e., 1, value is given when the PV is equal to the RV. Second, a zero reward "region" for all the remaining prediction states. What follows is that the less the predicted error is, the closer the reward approaches to 1. Having that said, we define the reward function as shown in formula (9).

4) Action Selection, Q-Table Update and State Transition: There are two strategies for the selection of actions, namely, *exploration* and *exploitation*. Exploration is the strategy used to collect more information of the environment, while exploitation is applied to make the best decision based on the available information. Exploration usually selects an action in random, whereas the exploitation applies a greedy policy towards an optimal choice. Even though the random policy is evidently suboptimal, still it enables to update the learning experience and adjust the Q-value to any changes in the environment.

In order to achieve a desired trade-off between the two strategies, we implement an *ϵ-greedy policy* and take each consequent action based on formula (10). The formula implies the generation of a pseudo-random number $0 \leq n \leq 1$ that is compared to the ϵ-greedy value. The latter is originally set to 0.5 and it gets decreased with a *decay factor* of 0.9 for each consequent action. This way, AREL3P starts with applying either of the two strategies with an equal probability, but then gradually increases (resp., decreases) the chances of taking experienced-based (resp., random-based) actions due to the ϵ-greedy policy.

The reward function (9) is used to yield an immediate reward and the Q-table gets updated according to formula (5). Finally, the state s_t changes to a new state s_{t+1} with the new coming data d_{n+1}.

$$R_a(s, s') = \begin{cases} 1 - |\frac{PV-RV}{0.1 \cdot RV}|, & \text{if } |PV - RV| \leq 0.1 \cdot RV; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$a = \begin{cases} \text{rand } A(s), & \text{if n } < \epsilon - \text{greedy}; \\ \arg\max_a Q(s_t, a), & \text{otherwise.} \end{cases} \tag{10}$$

5) Prediction Algorithm: AREL3P's prediction algorithm (see Algorithm (1)) starts by creating a new Q-table and an initial state. Then, it takes the input data *D* (see formula (8)) along with the initial system parameters, and performs predictions based on the ϵ-greedy policy. In order to maintain the prediction accuracy in the dynamic environment, Q-learning keeps updating its knowledge from the previous experience in the changing states, actions and rewards.

Algorithm 1 Prediction Algorithm of AREL3P

if Initialization **then**
  if not exist Q-table *Q(s, a)* **then**
    Create Q-table *Q(s, a)*
  Initialize state to s_0
  Initialize parameters ϵ, α, γ and *decay*
while True **do**
  //Training or predicting process
  if d_t is not null or VNF request **then**
    Take d_t
    //Choose an action using $\epsilon-$*greedy*
    if *rand*(0, 1) $< \epsilon-$*greedy* **then**
      $a_t = \text{rand } A(s)$
    else
      $a_t = \arg\max_a Q(s_t, a)$
    //Calculate the reward from the action applying
    ϵ *intensive band*
    if $|PV - RV| \leq 0.1 \cdot RV$ **then**
      $R_a(s, s') = 1 - |\frac{PV-RV}{0.1 \cdot RV}|$
    else
      $R_a(s, s') = 0$
    //Update Q-table
    $Q(s_t, a_t) = Q(s_t, a_t) +$
      $\alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
    //Use the value of the selected action
    $T_{predicted} = a_t$
end while

IV. System Evaluation and Validation

A. Testbed Architecture

Our testbed, illustrated in Figure 4, is deployed at the University of Bristol UK site of the 5GinFIRE[8] infrastructure. It is a multi-site 5G VNF ecosystem located at the UK, Spain, Portugal, Poland and Greece. The NFVI is composed of one controller/Compute Node (CN) and three CNs connected via Ethernet, following the equipment specifications summarized below:

- *Sensor:* 360°camera model Ricoh Theta V.
- *Source Node:* Raspi model 3B running Raspbian Jessi.
- *Compute/Controller Node 1:* CORSAIR ONE PRO, INTEL I7-7700K, 8-core processor, 16 GB RAM, 800 GB HD, running Ubuntu 16.04, KVM, Openstack Queen (Controller and Compute), OSM MANO release 4.
- *CN2:* Same as Node 1, except from: 80 GB HD, Openstack Queen (Compute).
- *CN3 & CN4:* IBM x3455, AMD Opteron, 4-core processor, 8 GB RAM, 70 GB HD, running Ubuntu 16.04, KVM, Openstack Queen (Compute).

[6]Note that this is four times higher than the average recorded T_{tot} (0.5 second) in all of the scenarios we run in our testbed.

[7]The notion of ϵ is used in several SL regressors such as Support Vector Regression (SVR) to predict continuous variables. The default value for these regressors is equal to 10% [11].

[8]https://5ginfire.eu/university-of-bristol-5g-testbed/

TABLE II
OPENSTACK NOVA [12]: DEFAULT FILTER SCHEDULER

ID	Filter	Description
1	RetryFilter	Filters out hosts that fail to response to a service request
2	AvailabilityZoneFilter	Filters hosts by availability zone according to the request
3	RamFilter	Places instances only in hosts that have sufficient ammount of RAM available
4	DiskFilter	Places instances only in hosts that have sufficient DISK space available
5	ComputeFilter	Approves all active CNs
6	ComputeCapabilitiesFilter	Matches compute capabilities against properties defined in extra specs for an instance type
7	ImagePropertiesFilter	Place instances only in hosts that can support the particular image properties contained in the instance
8	ServerGroupAntiAffinityFilter	Guarantees that each instance in a group is placed on a different host
9	ServerGroupAffinityFilter	Guarantees that an instance is placed inside a set of group hosts

TABLE III
DATASET FOR BENCHMARK SCHEMES

ID	Input Parameters	Description	Range/Unit	Source	Output (Predicted values)
1	Lo	Normalized CPU load	0 - 1	Raspi, PNs, ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
2	Mm	Memory utilization	0 - 100%	Raspi, PNs, ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
3	$Vcpu$	Number of cores used by virtual machine	Counting number	PNs	T_{rcv}, T_{p}
4	Bw	Bandwidth of the link	Megabit per second (Mbps)	Raspi, PNs, ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
5	Nt	Network utilization of the link	0 - 100%	Raspi, PNs, ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
6	Lt	Latency between two nodes	Millisecond (ms)	Raspi-PNs, PNs-ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
7	Ls	Packet loss between two nodes	0 - 100%	Raspi-PNs, PNs-ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}
8	Jt	Jitter between two nodes	Millisecond (ms)	Raspi-PNs, PNs-ENs	T_{trans}, T_{rcv}, T_{p}, T_{req}

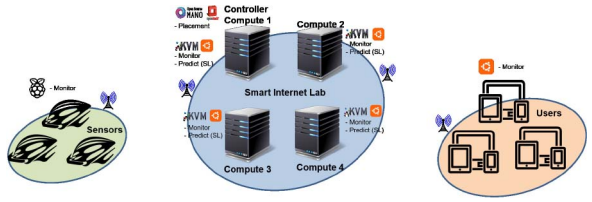

Fig. 4. Testbed of the experiment.

- *End-user Node:* Laptop model Acer Aspire VX15, running Ubuntu 16.04.
- *VNF video transcoder:* Virtual Display Unit (VDU) of a 2-core processor, 3 GB RAM, 4 GB HD, running Ubuntu 16.04.

B. Models for VNF Placement

Regarding our scheme, OSM MANO uses selects the host with the minimum predicted T_{tot} value by AREL3P. In addition, AREL3P applies a "*no overload*" mechanism that prevent a CN from overloading after placing the VNF. Using formula (11) below, it calculates the future load of the candidate CN after placing the VNF and rejects the node in case that its future load is above a threshold:

$$Load_{predict} = \frac{(Load_{VNF} + Load_{node})}{Core_{num}} \times 100, \qquad (11)$$

where $Load_{predict}$ is the predicted load percentage of a node after placing the VNF, $Load_{VNF}$ is the current load of the VNF, $Load_{node}$ is the current load of the node, and $Core_{num}$ is the number of the CPU cores of the node.

In what follows, we explain in detail the benchmark models we used for our evaluation purposes.

1) Non ML-Based Benchmarks: We use two different non ML-based benchmark orchestration schemes, namely (i) "Traditional" and (ii) "Random", against OSM MANO powered by AREL3P. While Random simply selects one of the three CNs at random, Traditional is more complex by adapting the existing VNF placement approach of the NOVA filter scheduler [12]. The filter scheduler uses two strategies: "*filtering*" and "*weighting*", respectively. Filters are essentially sets of rules, as briefly described in Table II, which define the resources and capabilities of a CN for hosting a VNF. The weighting strategy, on the other hand, applies weight to all filters to define their degree of influence to the final placement decision.

2) ML-Based Benchmark Models: We developed and trained five SL model algorithms. The purpose of these models is dual: first, to study their ability to generalize, as discussed in Section II. The five SL model algorithms are: Decision Tree (DT) [13], Random Forest (RF) [14], Linear Regression (LR) [15], SVR [16], K-Nearest Neighbors Regression (KNNR) [17].

To predict T_{tot}, the SL models take eight input parameters (namely, IDs 1-8 in Table III) and predict the time spent in each process separately. Predictions are computed using the general equations for SL algorithms shown in formulas (12) to (15):

$$T_{\text{trans}} = f\Big\{(Lo, Mm)_{\text{Rp}}, (Bw, Nt, Lt, Ls, Jt)_{\text{Rp-PN}}, Lo_{\text{PN}}\Big\} \qquad (12)$$

$$T_{\text{rcv}} = f\Big\{(Lo, Mm, Vcpu)_{\text{PN}}, (Bw, Nt, Lt, Ls, Jt)_{\text{Rp-PN}}\Big\} \qquad (13)$$

$$T_{\text{p}} = f\big\{(Lo, Mm, Vcpu)_{\text{PN}}, (Bw, Nt, Lt, Ls, Jt)_{\text{EN-PN}}\big\} \qquad (14)$$

$$T_{\text{req}} = f\big\{(Lo, Mm)_{\text{EN}}, (Bw, Nt, Lt, Ls, Jt)_{\text{EN-PN}}, Lo_{\text{PN}}\big\} \qquad (15)$$

where: $f\{x_1, \ldots, x_n\}$ is the function of corresponding SL model with the input parameters x_1 to x_n.

Then, the best SL model for VNF placement is used as benchmarks against AREL3P

3) Training and Testing Setup: For the sake of a proper validation of our solution, we evaluate and compare AREL3P against the five SL benchmarks introduced in Section IV-B2 based on (i) the accuracy of T_{tot} predictions and (ii) the respective efficiency of OSM MANO placement decisions under dynamic network conditions in two corresponding scenarios listed below. Note, that we adapt a high node churn scenario where nodes can arbitrarily join and depart at any time. This means that newly-joined CNs are deployed with (i) an RL model that is not yet trained or (ii) an SL model trained for another previously existing CN.

a) Scenario 1 (accuracy): The objective of this scenario is to evaluate the accuracy of predictions for T_{tot} by AREL3P against those by the SL models running at the nodes they were trained for as well as the newly-joined nodes that they were not trained for. We use the same dataset for training both AREL3P and the five benchmarks, which comprises of over 300K instances collected from low, medium and high Key Performance Indicators (KPIs) (load, latency, packet loss and bandwidth) at all servers. Regarding the SL banchmarks, in particular, we use *grid search* for selecting the set of optimal hyperparameters.

We deploy different AREL3P instances in each node irrespective of their specifications presented in Section IV-A. The model goes through an exhaustive training phase that involves a 100 iterations with the initial Q-learning parameters set to: $< \epsilon = 0.1; \alpha = 0.1; \gamma = 0.9; \epsilon - greedy = 0.5; decay = 0.9 >$. Note that we repeat the training and evaluation of our model x30 in order to increase confidence to our recorded results.

Finally, to train, test and validate the benchmark models we use 10-fold cross-validation.

b) Scenario 2 (VNF placement efficiency): This scenario is designed to assess the quality of VNF placements after T_{tot} predictions, thus providing a more holistic approach to validating AREL3P than simply focusing on the (raw) accuracy of T_{tot} predictions. We set CN 4 as "overloaded" and have it to trigger a VNF request message upon its CPU load reaching to 80% of its maximum capacity. We choose this threshold value after [18], which shows that response times of running processes start to increase exponentially when the load exceeds 70% and become critical when the load reaches to 80%. The rest of the CNs are considered as candidates for placing the VNFs irrespective of their different specifications and current statuses. Furthermore, we engage into 27 different scenario repeats (see Table IV) with varying load profiles in candidate nodes. We define as "Low" load that status of using 0-30% of the node's maximum CPU capacity, a "Medium" load for using 30-70% and a "High" load for exceeding 70% of the node's capacity. To demonstrate the adaptability of AREL3P, each instance in each scenario updates its model from the first test by using 10%[9] of total samples (30000:300000). Then the VNF placement test is performed using these different approaches: a Traditional and a Random one, as well as AREL3P in the *dynamic* environment.

[9]SQL Server updates a large database using sample sizes between 10-30% [19]. As fast deployment is preferable in a dynamic network, we select the minimum possible value (10%) for model training, which accounts for 8 hours extracted out of our data collection.

TABLE IV
TESTING SCENARIOS

Repeat	CN 1	CN 2	CN 3
1	Low	Low	Low
2	Low	Low	Medium
3	Low	Low	High
4	Low	Medium	Low
5	Low	Medium	Medium
6	Low	Medium	High
7	Low	High	Low
8	Low	High	Medium
9	Low	High	High
10	Medium	Low	Low
11	Medium	Low	Medium
12	Medium	Low	High
13	Medium	Medium	Low
14	Medium	Medium	Medium
15	Medium	Medium	High
16	Medium	High	Low
17	Medium	High	Medium
18	Medium	High	High
19	High	Low	Low
20	High	Low	Medium
21	High	Low	High
22	High	Medium	Low
23	High	Medium	Medium
24	High	Medium	High
25	High	High	Low
26	High	High	Medium
27	High	High	High

Finally, the results from all approaches are compared with one another and Benchmark.[10]

C. Evaluation Results

The evaluation results refer to *accuracy scores* with values denoting the average score out of all iterations. These scores show the percentage of the prediction data that match the real data. Regarding the adaptability and native run scenario of Section IV-C1, in particular, the results are compared with R-squared (R2) scores. R2 is a statistical tool that measures "Goodness-of-Fit", the efficiency of the prediction of a regression model to the real data points. The R2 formula is defined as:

$$1 - \frac{SS_{reg}}{SS_{tot}}, \tag{16}$$

where SS_{reg} is the regression sum of squares and SS_{tot} is the total sum of squares. R2 values range between 0 and 1, with 1 denoting the best score. It can become negative if the regression model does not fit the data at all. Note that we convert R2 scores to percent to simplify the analysis.

1) Scenario 1 (Accuracy): The SL models were trained and evaluated based on real 60-hour data collected from our testbed over a period of one week and in two scenarios; namely, "*Native run*" and "*Adaptability*". Note that the data came from a variety of processing and network statuses, as shown in Table III. For the case of the "Native run" scenario, all the models were deployed to the nodes that they where trained for, while for the "Adaptability" scenario the trained models were deployed to nodes with different specifications.

Figure 5 portrays our relevant evaluation results, showing that SL models can be highly accurate for "Native run".

[10]The optimal placement results from Supervised Learning model in static environment.

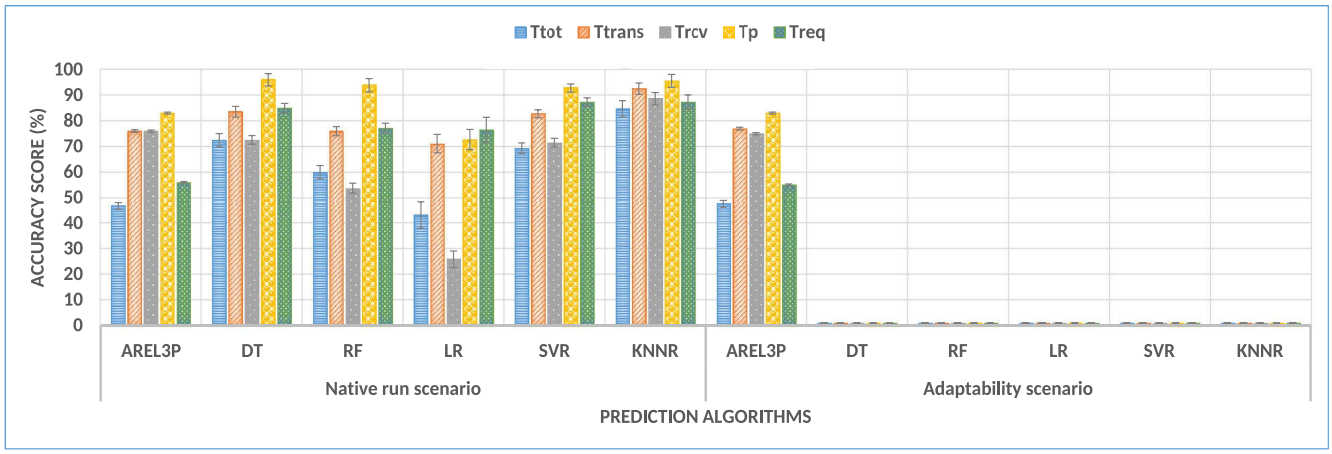

Fig. 5. Accuracy scores for T_{tot} and its details components T_{trans}, T_{rcv}, T_{p}, T_{req} for AREL3P and all SL models in both the "Native run" and "Adaptability" scenarios.

But on the very contrary, the same models exhibit a complete lack of adaptability. This is a major drawback caused by the fact that the SL models are tailored to specific nodes with given specifications (e.g., CPU capacity, demand pattern or network delay). The overall resilience superiority of AREL3P is verified by the accuracy score for T_{tot} during the "Adaptability" scenario, which is approximately equal to 45% contrary to the zero accuracy of the SL models. AREL3P can dynamically *adopt* to network conditions irrespective of node specifications due to the continuous online training nature of RL even after deployment. This is not the case for the offline trained SL models, which remain static after deployment and lack resilience to "dataset shifting" [20]. For completeness, the graph also shows the accuracy scores of AREL3P in the "Adaptability" scenario per individual components of T_{tot}, namely: 78% regarding T_{trans}; 75% regarding T_{rcv}; 82% regarding T_{p}; and 58% regarding T_{req}. As it can be easily observed, the corresponding "Native run" accuracy scores for AREL3P T_{tot} and its components exhibit little differences compared to their "Adaptability" scenario counterparts. This reveals a solid performance behaviour.

In further, the graphs of Figure 6 portray the prediction accuracy scores achieved by AREL3P for each task during an exhaustive test of over 100 iterations. Notice that all minimum values that we refer to next, correspond to accuracy scores per task after iteration 1 (resp., maximum after approximately 30 iterations). Specifically, Graph 6(a) shows that the minimum prediction accuracy score for T_{trans} is 10% (resp., maximum is approximately 85%). Regarding T_{rcv} in Graph 6(b), the minimum value is 7% (resp., maximum is 82%). Graph 6(c) denotes that the minimum prediction accuracy for T_p is 10% (resp., maximum is 90%). Last, the minimum prediction accuracy for T_{req} in Graph 6(d) is 7% (resp., maximum is 65%).

Commenting more on these results, we see that AREL3P's accuracy performance under dynamic conditions is quite similar in all graphs during iterations 1 to 20, where we observe a large increase in accuracy scores for all tasks. During iterations 20 to 30, AREL3P's accuracy performance continues to increase in all graphs, albeit at a slower rate, and remains at a maximum score between approximately iterations 30-40 (30-33 only for T_{req} in Graph 6(d)). This means that the model stabilizes during these iterations due to the suitable ϵ-greedy values. Finally, for the rest of the iterations after iteration 40, we observe a high performance fluctuation and gradual decline in all graphs, as well as an increase of confidence intervals. This implies that AREL3P's accuracy declines after approximately iterations 30 to 40. We elaborate on the underlying reasons for that decline when analyzing the results of Figure 7 next.

The results of Graph 7(a) demonstrate the prediction accuracy of our model for T_{tot} after each iteration. The highest accuracy score achieved by AREL3P is approximately 43% after iteration 29. Note that this score is less compared to the scores in the graphs of Figure 6, which is due to the fact that T_{tot} corresponds to predictions made for all end-to-end devices of our setup in the context of the SCS use case by using formula (7) to estimate the corresponding T_{tot} predictions. These added number of predictions causes to accumulate prediction errors for T_{tot}.

Commenting more on the exhibited performance pattern, we observe a significant accuracy decrease after iteration 29. This pattern can be better understood by the illustrated *exploration-exploitation* Graph 7(b). The Y-axis values correspond to the gradually changing probability ratio ((see details in Section III-B4)) between using exploration or exploitation ($\epsilon-$*greedy* with *decay factor*) in our underlying Q-learning model. Notice that the optimal scores concentrate around iterations 15-30. During a first phase (iterations 1 to 20) where accuracy scores raise rapidly in Graph 7(a), the probability of the exploration strategy drops from 45% to 8% (resp., it grows from 55% to 92% for the exploitation strategy). Evidently, a higher exploration probability has a negative impact on AREL3P's prediction accuracy, as this strategy picks actions randomly. On the contrary, a higher exploitation probability improves accuracy since predictions are made after the trained Q-learning model.

Nevertheless, if exploitation probability reaches too high (i.e., 96% - 100%), this does not improve the AREL3P

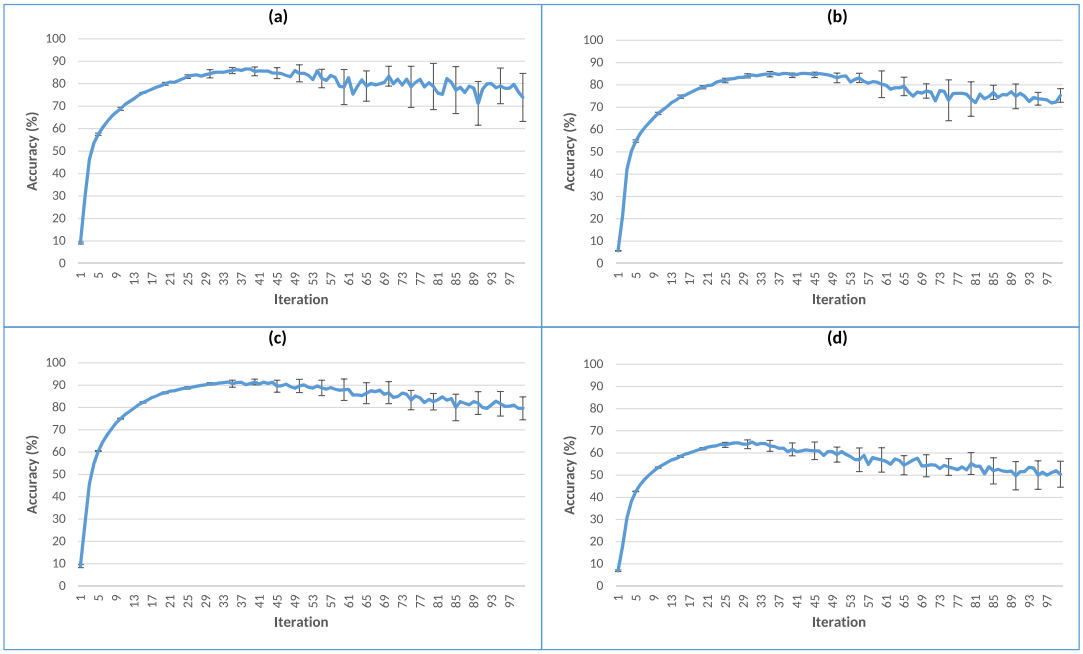

Fig. 6. Accuracy scores against testing iterations for (a) T_{trans}, (b) T_{rcv}, (c) T_{p} and (d) T_{req}.

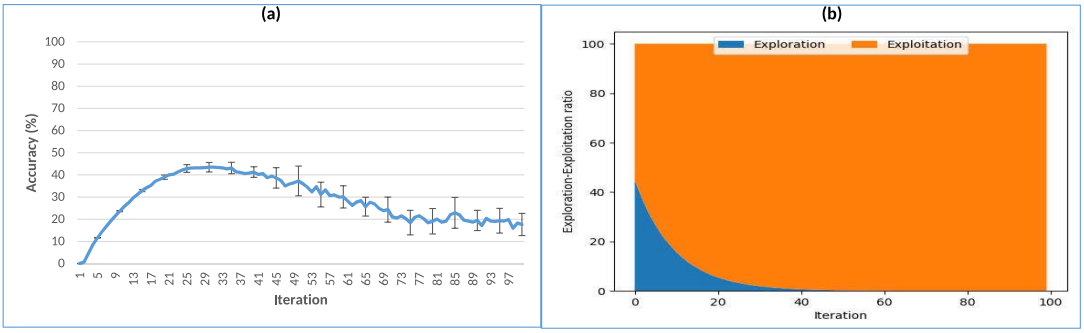

Fig. 7. (a) Accuracy scores of T_{tot} and (b) Exploration-Exploitation ratio.

prediction accuracy as demonstrated in a second phase of iterations (iterations 20 to 40). Having a correspondingly very low exploration ratio (i.e., 4% - 0%, particularly during iterations 35 to 40), leads to AREL3P being unable to adopt sufficiently to new samples, hence impacting our model's adaptability and resilience to changes in the network environment. The latter provides also an explanation on why the accuracy scores remain steady at the beginning of phase two and then decrease gradually up until the end of this phase.

Finally, our model uses only the exploitation strategy during a third and last phase (iterations 40 to 100). Without exploration, all accuracy score results inevitably drop and exhibit a decreased confidence due to a high statistical deviation. This proves that the AREL3P prediction model *needs to be updated at a certain level to maintain its good prediction accuracy*. In conclusion, the optimal results w.r.t. T_{tot} accuracy are observed during iterations 25 to 33, corresponding to a probability of 94-96% for using the exploitation strategy (i.e., 6-4% for using exploration, respectively).

As an overall conclusion out of all the result in "Scenario 1 (Accuracy)", AREL3P predictions of T_{tot} are *resilient to dynamic environment conditions*, hence achieving accuracy levels between *40-45%*, unlike the zero accuracy and corresponding lack of resilience of SL models. Also, we can conclude that AREL3P requires both exploitation and exploration, with the optimum ratio between the two being approximately 94-96%:6-4%.

2) Scenario 2 (VNF Placement Efficiency): The results presented below denote that AREL3P leads to good placement decisions, nearly as good as the ones made by the best benchmark predictions out of all SL model options (therefore, notated as "Best Benchmark"). Placement decisions after input from AREL3P may not always lead to an optimal VNF placement, however the selected locations are shown to yield a VNF performance that is very close to optimal.

Figure 8 depicts the performance of a new VNF after its placement. Recall that lower T_{tot} values indicate a better performance. As seen, AREL3P and benchmark results are very similar. This figure also shows that the AREL3P outperforms both the Traditional method and the Random method for placing VNFs in many scenarios on the X-axis. This is because the two latter strategies do not consider the VNF performance.

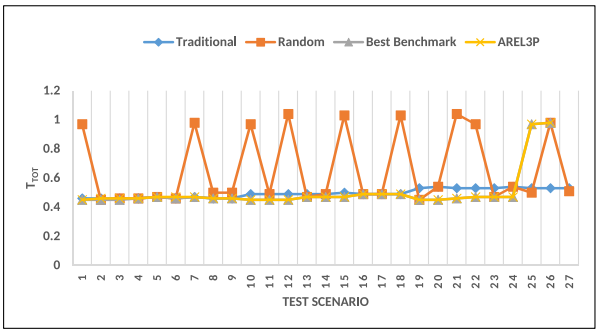

Fig. 8. T_{tot} of the new VNF after instantiation.

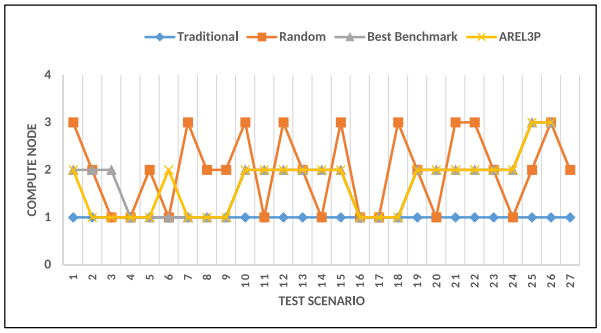

Fig. 9. Node selection from Traditional, Random, Benchmark and AREL3P approaches.

As for the Traditional placement method achieving optimal results in some scenarios (25 - 27), the latter comes at the (undesired) cost of overloading the placement node afterward (see Figure 10).

Moving forward to the results of Figure 9, the graph demonstrates the placement performance of all four VNF placement algorithms in a total of 27 scenarios. Notice that the Traditional approach always places a new VNF at the CN 1. This is because its decisions are based on the filters scheduler, where the resources such as memory (RamFilter) and disk space (DiskFilter) pose the highest influence among filters. In this testbed, CN 1 has always the most resources because we only change the load during the test. Considering the RamFilter, the memories of CN 1 and CN 2 are the same or greater size than CN 3. Furthermore, regarding the DiskFilter, the CN 1 possesses the largest disk capacity, followed by CN 2 and CN 3 respectively.

The results from Best Benchmark and AREL3P are alike. Both models mostly select CN 2 to place a new VNF, followed by choosing CN 1 and CN 3, but none of them selects any CN for a new VNF in the scenario 27. The placement decisions of both models differ in the scenarios 1 - 6. The Benchmark model selects CN 2 during scenarios 1-3 and CN 1 during scenarios 4-6 because VNFs' performances are best in those nodes with their given loads. In contrast, AREL3P chooses CN 2 in scenarios 1, 6 and CN 1 in scenarios 2-5. AREL3P's placement performance is poorer in these scenarios because the AREL3P prediction model has around 50% accuracy (see Figure 7) causing errors on AREL3P's predictions and, eventually, leading to non-optimal VNF placements. However, these errors are "acceptable" since the difference of T_{tot} between

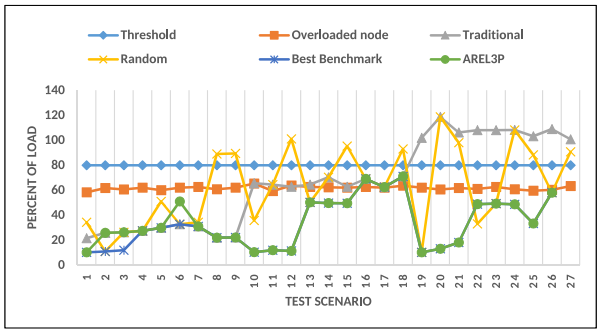

Fig. 10. Load in the overloaded node and winner node after the instantiation.

CN 1 (running at a low load) and CN 2 (running at low/medium load) are *negligible*, as illustrated in Figure 8.

In scenarios 7-9, the load for CN 2 is high, whereas the load remains low for CN 1. Consequently, the T_{tot} predictions for CN 2 are greater than those for CN 1. Both models select CN 1 for a new VNF.

In scenarios 10-15, the load of CN 1 is medium, while the load for CN 2 is either low or medium. As a result, predictions for CN 2 are lower than those for CN 1. Both models choose CN 2 to place the new VNF.

In the scenarios 16-18, the load of CN 2 raises to high, while for CN 1 it remains at a medium level. As a result, predictions for CN 1 are lower than those from for CN 2, causing to instantiate the new VNFs at CN 1 by both models.

In further, the load of CN 1 rises to high for scenarios 19-27. CN 2 gets selected by both the Best Benchmark and AREL3P for placing a new VNF during scenarios 19-24 because the predicted values are the lowest. However, in scenarios 25 and 26, both models lead to selecting CN 3 for placing the VNFs. This is due to the fact that CN 1 and CN 2 get rejected due to the "no overload" check mechanism (see Section IV-B).

Last, the loads in all nodes are high for scenario 27. In this case, no node passes the "no overload" check and the VNF placement does not occur at all.

Analyzing the impact of load further, Figure 10 portrays the loads in the overloaded node and the winner node that gets the placement decision. In the overloaded node, its load after the placement decreases to below the threshold (80%) in all scenarios. However, the load of the winner nodes' remain under the threshold for Best Benchmark and AREL3P, whereas when these winner nodes apply Traditional or Random they become overloaded in many scenarios. The latter indicates that Best Benchmark and AREL3P can better handle the impact on (over)load at nodes after placement decisions.

Finally, we compare the Quality of Decision (QoD) of all four methods. QoD ranges from 0 to 3 based on two criteria: "*Best* T_{tot} *(Bt)*" and "*Not overloaded (No)*". For the first assessment, the Bt will give one point to any placements at the node with the least T_{tot}. For the second assessment, as overloading a node is undesirable, No gives a reward of two points when the winner nodes do not become overloaded after the placement. The other results of the decision are *not* be rewarded. Figure 11 depicts QoD of Traditional, Random, Benchmark and AREL3P approaches. The Benchmark model receives the best QoD scores in 26 out of 27 scenarios. The last

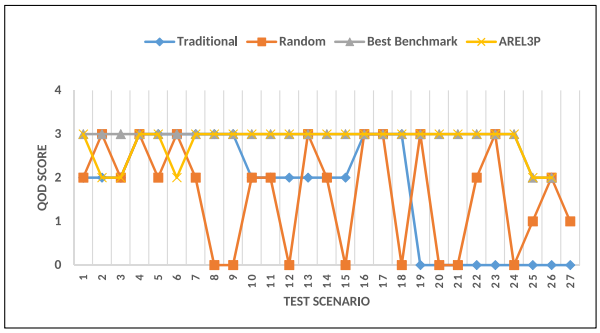

Fig. 11. QoD scores of Traditional, Random, Benchmark and AREL3P approaches.

scenario is not evaluated because no placement occurs. The AREL3P approach is a runner-up with the best QoD scores in 23 of 27 scenarios. Finally, the Random method and the Traditional method prove to be the worst strategies with the best QoD scores in 10 and 9 out of 27 scenarios, respectively. From the results above, it is proved that the AREL3P can achieve a performance that is near to the one by the Best Benchmark. Therefore, AREL3P performs well w.r.t. VNF placements for latency-sensitive and/or e2e applications.

V. State of the Art

A. Non ML-Based Approaches

In the context of Linear Programming (LP), Cohen *et al.* [21] use LP-relaxation for VNF chain placement in inter-Data Center Network (DSN) environments with bicriteria approximation factors. This solution, however, violates size constraints by a constant factor. Bhamare *et al.* [22] propose an Integer Linear Programming (ILP) model that minimizes both traffic and total response time between in multi-cloud environment. Their model also considers other constrains such as total deployment costs and Service Level Agreements (SLAs). Using ILP again, Bari *et al.* [23] apply a model that minimizes the operational costs and increases utilization by determining the number of necessary VNFs and their locations.

Focusing on Mixed Integer Linear Programming (MILP), Rocha and Verdi [24] present a traffic-aware model for VM placement in DSNs, which places VMs w.r.t. traffic patterns. Unlike that, Zhao *et al.* [25] combine a MILP model to an efficient heuristic that is based on Lagrange's relaxation decomposition to optimize VMs placement and the overall topology considering the dynamic traffic conditions of DSNs. Taking a different approach, Mehraghdam *et al.* [26] propose a context-free language for specifying VNF chains and then provide a Mixed Integer Quadratically Constrained Program (MIQCP) formulation for VNF chain placement.

There is, also, substantial work in the literature outside the context of LP based on a variety of different algorithmic and heuristic approaches. Even *et al.* [27] propose a randomized approximation algorithm for path computation and function placement. Their solution uses two optimization problems: (i) "path computation", which focuses on packet forwarding; and (ii) "function mapping", which focuses on load balancing.

The work of Xu *et al.* [28] proposes an algorithm for the VNF placement that leverages the trade-off between energy consumption and the probability of SLA violation.

Tajiki *et al.* [29] apply a mathematical approach to predict traffic and, correspondingly, reallocate resources in SDN networks in an effort to reduce the total packet loss while increasing network throughput. The works of [30], [31] use dynamic congestion pricing models for distributed resource allocation. Both dynamic prices and dynamic user mobility prediction information per node are used to trade the cost of a local resource for reduced e2e data transfer delay. Lange *et al.* [32] apply Pareto-based heuristic approaches to optimize VNF placement according to various metrics including the latency of all devices, their resilience against both node and link failures, and load balancing.

Regarding edge computing solutions, Aral and Ovatman [33] design and present a replica placement algorithm that targets latency improvement and cost reduction w.r.t. the geographical locality of data during the dissemination process. Unlike that, the work of Yang *et al.* [34] proposes a dynamic resource allocation framework based on a fast incremental allocation heuristic that dynamically performs resource allocation and a periodic re-optimization algorithm that adjusts resources to maintain a near-optimal edge operational cost.

Last, Clayman *et al.* [35] introduce three simple placement algorithms based on different criteria, namely, (i) "Least Used Host", (ii) "N at a Time in a Host", and (iii) "Least Busy Host". The first algorithm selects the host that has the least number of virtual routers. The second algorithm allocates N routers to a single host at once. Last, the third algorithm chooses the host with the least virtual network traffic.

As an overall remark, the majority of non ML-based approaches in the literature do not considered e2e delay. Instead, they focus on resource allocation and/or VNF placement after local-host system measurements. Therefore, they can not address requests sensitive to e2e delay.

B. ML-Based Approaches

MLN has been studied in various aspects such as prediction, classification or policy selection. However, none of them consider the e2e delay.

The work of [36] by Jmila *et al.* adapts SVR to estimate VNF resource requirements by predicting the CPU demand of incoming traffic. The authors compare their results against those of an Artificial Neural Network (ANN) and shown that SVR outperformes ANN in terms of both accuracy and stability.

Shi *et al.* [37] proposed a MDP method to dynamically allocate cloud resources for VNFs, followed by applying Bayesian learning to predict the probability of resource reliability.

Rankothge *et al.* [6] propose a genetic algorithm for VNF chain placement comprising of two modules: New function provisioning and Scaling out/in. The earlier is responsible for the resource allocation while the latter assigns a new set of NFs and paths to satisfy the current networking demand.

The work of [38] studies a proactive ML-based approach for VNF auto-scaling in response to dynamic traffic changes. The authors evaluate seven different SL models for classifying

"Positive" and "Negative" samples of traffic-load measurement data, showing that ML improves QoS while saving significant cost for both network owners and leasers over a time-series moving average prediction approach. Also, Xu *et al.* [7] implement a combination of MDP and learning algorithms to optimize the policy of dynamic workload offloading in both cloud and edge servers.

Finally, there are only a few pieces of (recent) work [2], [3], [5] in the literature that use RL. Unlike these works, ours uses specifically Q-Learning with certain advantages discussed in Section II-A, focusing on end-to-end application level performance predictions that are passed to well-established orchestrators for taking efficient VNF placement decisions. Tang *et al.* [2] tackle the auto-scaling problem by combining MDP and RL. They apply MDP to find a tradeoff between the achieved level of QoS and the VNF resource consumption. Then, they use RL to define a threshold based policy. Their results illustrate that this approach outperforms both a static threshold and a voting policy methods. Chen *et al.* [3] optimize two Deep RL Agents (RLAs), namely, a Short flow RLA (sRLA) and a Long flow RLA (lRLA). The sRLA optimizes the threshold for Multi-Level Feedback Queueing (MLFQ) by applying Deep Deterministic Policy Gradient (DDPG), while the lRLA determines the rates, the routes and the priorities for long flows by implementing a flow scheduler. Mijumbi *et al.* [4] propose a multi-agent learning algorithm for virtual network resource management. These agents learn an optimal policy from online feedback and dynamically allocate network resources to virtual nodes and links. Xiao *et al.* [5] propose NFVdeep to automatically deploy Service Function Chaining (SFC)s for requests with different QoS requirements. This work applies an adaptive, online, deep reinforcement learning approach along with a serialization-and-backtracking method and a policy gradient based method to handle SFCs deployment in the real-time network with variations and various service requests.

VI. Conclusion and Future Work

This article presents an Adapted REinforcement Learning VNF Performance Prediction module for Autonomous VNF Placement (AREL3P) to enhance MANagement and Orchestration (MANO) systems. Taking a different approach from other solutions in the literature based on Supervised Learning (SL), AREL3P adapts a particular type of RL, namely Q-learning. To the best of our knowledge, our effort is the first one in the literature to leverage RL predictions for the purposes of ZSM. As validated by our meticulous performance evaluation over a *realistic* testbed environment and use case setup, our adapted Q-learning scheme exhibits a better tolerance to network dynamics than SL-based models. This is due to the fact that Q-learning is an online learning technique that allows to update the learning model in operation, thus overcoming most of the raised feasibility and adaptability concerns faced by SL-based models. In general, our results show that AREL3P yields more accurate VNF performance predictions using end-to-end service level information. The latter constitutes another novelty compared to most existing works in the literature on VNF placement including non ML-based ones. Specifically, AREL3P can predict end-to-end service level VNF performance with a 40-45% higher accuracy compared to SL models. We also conclude that the optimal exploration-exploitation ratio for training our Q-learning model is approximately 6-4%:94-96%. In addition, our VNF placement tests prove that AREL3P receives the best Quality of-Decision scores in 23 of 26 investigated scenarios w.r.t. to our adapted SCS use case.

For future work, we will focus on applying ML at the level of management and orchestration such as for the purposes of scheduling or profiling. We also plan to study and to integrate multiple levels of ML models. We believe that this can play a key role in improving the efficiency of VNF placement decisions, especially for end-to-end or delay-sensitive VNFs.

References

[1] M. Jie *et al.*, "Zero-touch network and service management," Deutsche Telekom, Bonn, Germany, White Paper, Dec. 2017. Accessed: Apr. 13, 2018. [Online]. Available: https://portal.etsi.org/TBSiteMap/ZSM/OperatorWhitePaper

[2] P. Tang, F. Li, W. Zhou, W. Hu, and L. Yang, "Efficient auto-scaling approach in the telco cloud using self-learning algorithm," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[3] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTo: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. Conf. ACM Special Interest Group Data Commun.*, New York, NY, USA, 2018, pp. 191–205. [Online]. Available: http://doi.acm.org/10.1145/3230543.3230551

[4] R. Mijumbi, J. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Krakow, Poland, 2014, pp. 1–9.

[5] Y. Xiao *et al.*, "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proc. Int. Symp. Qual. Serv.*, New York, NY, USA, 2019, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3326285.3329056

[6] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, May 2015, pp. 89–97.

[7] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[8] *University of Bristol 5G Testbed 5GinFIRE*. Accessed: Jan. 14, 2019. [Online]. Available: https://5ginfire.eu/university-of-bristol-5g-testbed/

[9] *Layered Realities—Smart City Safety Faculty of Engineering University of Bristol*, Uiniv. Bristol, Bristol, U.K. Accessed: May 4, 2018. [Online]. Available: http://www.bristol.ac.uk/engineering/research/smart/events/layered-realities-weekend/layered-realities—smart-city-safety/

[10] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996. [Online]. Available: https://link.springer.com/article/10.1007/BF00116900

[11] *sklearn.svm.SVR Scikit-Learn 0.21.1 Documentation*. Accessed on: May 20, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[12] *OpenStack Docs: Compute Schedulers*. Accessed: Dec. 20, 2018. [Online]. Available: https://docs.openstack.org/ocata/config-reference/compute/schedulers.html

[13] *Decision Trees*. Accessed: Oct. 18, 2018. [Online]. Available: http://scikit-learn.org/stable/modules/tree.html

[14] *A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python)*. Apr. 2016. Accessed: Feb. 11, 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

[15] *Multiple Linear Regression*. Accessed: Oct. 18, 2018. [Online]. Available: http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm

[16] *Understanding Support Vector Machine Regression*. Accessed: Oct. 18, 2018. [Online]. Available: https://uk.mathworks.com/help/stats/understanding-support-vector-machine-regression.html
[17] K. Yu, L. Ji, and X. Zhang, "Kernel nearest-neighbor algorithm," *Neural Process. Lett.*, vol. 15, no. 2, pp. 147–156, Apr. 2002. [Online]. Available: https://doi.org/10.1023/A:1015244902967
[18] A. Svensson, "Dynamic alternation between load sharing algorithms," in *Proc. 25th Hawaii Int. Conf. Syst. Sci.*, vol. 1. Kauai, HI, USA, Jan. 1992, pp. 193–201.
[19] *SQL Server Statistics: Explained*. Accessed: May 23, 2019. [Online]. Available: https://blogs.msdn.microsoft.com/srgolla/2012/09/04/sql-server-statistics-explained/
[20] J. Quioñero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.
[21] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Kowloon, Hong Kong, Apr. 2015, pp. 1346–1354.
[22] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366417301901
[23] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Serv. Manag. (CNSM)*, Barcelona, Spain, Nov. 2015, pp. 50–56.
[24] L. A. Rocha and F. L. Verdi, "A network-aware optimization for VM placement," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl.*, Gwangiu, South Korea, Mar. 2015, pp. 619–625.
[25] Y. Zhao, Y. Huang, K. Chen, M. Yu, S. Wang, and D. Li, "Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks," *Comput. Netw.*, vol. 80, pp. 109–123, Apr. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861400468X
[26] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.
[27] G. Even, M. Rost, and S. Schmid, "An approximation algorithm for path computation and function placement in SDNs," in *Structural Information and Communication Complexity*. Cham, Switzerland: Springer, 2016, pp. 374–390.
[28] S. Xu, B. Fu, M. Chen, and L. Zhang, "An effective correlation-aware VM placement scheme for SLA violation reduction in data centers," in *Algorithms and Architectures for Parallel Processing*. Cham, Switzerland: Springer, 2015, pp. 617–626.
[29] M. M. Tajiki, B. Akbari, and N. Mokari, "QRTP:QoS-aware resource reallocation based on traffic prediction in software defined cloud networks," in *Proc. 8th Int. Symp. Telecommun. (IST)*, Tehran, Iran, Sep. 2016, pp. 527–532.
[30] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in *Proc. ACM 2nd Edition Inf. Centric Netw. Workshop (ICN'12)*, Helsinki, Finland, Aug. 2012, pp. 61–66. [Online]. Available: https://doi.org/10.1145/2342488.2342502
[31] X. Vasilakos, V. A. Siris, and G. C. Polyzos, "Addressing niche demand based on joint mobility prediction and content popularity caching," *Comput. Netw.*, vol. 110, pp. 306–323, Dec. 2016. [Online]. Available: https://doi.org/10.1016/j.comnet.2016.10.001
[32] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 12, no. 1, pp. 4–17, Mar. 2015.
[33] A. Aral and T. Ovatman, "A decentralized replica placement algorithm for edge computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 2, pp. 516–529, Jun. 2018.
[34] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 1, pp. 475–488, Mar. 2018. [Online]. Available: https://doi.org/10.1109/TNSM.2018.2790081
[35] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, Krakow, Poland, May 2014, pp. 1–9.
[36] H. Jmila, M. I. Khedher, and M. A. El Yacoubi, "Estimating VNF resource requirements using machine learning techniques," in *Neural Information Processing*. Cham, Switzerland: Springer, 2017, pp. 883–892.
[37] R. Shi *et al.*, "MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," in *Proc. IEEE Int. Conf. Serv. Comput.*, New York, NY, USA, Jun. 2015, pp. 65–73.
[38] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling VNFs using machine learning to improve QoS and reduce cost," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

Monchai Bunyakitanon received the B.S.E.E. degree from the Escuela Naval Militar, Pontevedra, Spain, in 2006, and the M.S.E.E. degree with an emphasis in telecommunication systems from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2014. He is currently pursuing the Ph.D. degree in electrical engineering at the University of Bristol, U.K. He is an Officer of the Royal Thai Navy and a member of the Smart Internet Lab. His research focus includes machine learning solutions for multiaccess edge computing and networking, network function virtualization, and software-defined networks.

Xenofon Vasilakos received the M.Sc. degree in parallel and distributed computer systems from Vrije Universiteit Amsterdam, and the Ph.D. degree in informatics from the Athens University of Economics and Business with a focus on information-centric networking architectures, protocols, and distributed solutions. He is currently a Research Fellow with the University of Bristol, Bristol, U.K., where he is a member of the Smart Internet Laboratory and the Technical Lead Researcher of the Zero Downtime Edge Application Mobility (MEC Mobility) project. He has participated in various EU and national funded research projects, such as 5GPPP SliceNet and the FIA award-winning FP7 project PURSUIT. His current research interests include 5G/B5G technologies with a focus on multiaccess edge computing based on cognition approaches inspired by machine learning models toward self-managed networks. He is also involved in the areas of Internet of Things, software-defined networking, network function virtualization, and network slicing in the context of 5G. He was a recipient of an Excellence Fellowship Grant from the French Government (LABoratoires d'EXcellence), and has received an accolade and awards for his academic performance from the Greek State Scholarship Foundation.

Reza Nejabati (Senior Member, IEEE) has written or coauthored over 200 peer reviewed papers and several standardization documents. His current area of research is in the field of disruptive new Internet technologies with focus on application of high-speed network technologies. He has a successful track record in working at the interface between optical networks and computer science, as well as between academia and industry. Throughout his research career, he has made important and pioneering contributions to the fields of optical networking, grid networking, data center networks, software defined networking, network virtualization, and network function virtualization.

Dimitra Simeonidou (Fellow, IEEE) is a Full Professor at the University of Bristol, where she is the Co-Director of the Bristol Digital Futures Institute and the Director of the Smart Internet Lab. She is increasingly working with Social Sciences on topics of digital transformation for society and businesses. She has been the Technical Architect and the CTO of the smart city project Bristol Is Open. She is currently leading the Bristol City/Region 5G urban pilots. She has been co-founder of two spin-out companies, the latest being the University of Bristol VC funded spin-out Zeetta Networks, http://www.zeetta.com, delivering SDN solutions for enterprise and emergency networks. She has authored and coauthored over 500 publications, numerous patents and several major contributions to standards. Her research is focusing in the fields of high performance networks, programmable networks, wireless-optical convergence, 5G/B5G, and smart city infrastructures. She is a fellow of the Royal Academy of Engineering and the Royal Society Wolfson Scholar.