Computer Networks 166 (2020) 106980

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks[☆]

Tejas Subramanya[a,*], Davit Harutyunyan[a], Roberto Riggio[a]

FBK CREATE-NET, Via Alla Cascata 56/C, Trento 38123, Italy

ARTICLE INFO

Article history:
Received 15 August 2019
Revised 29 October 2019
Accepted 4 November 2019
Available online 5 November 2019

Keywords:
Auto-scaling
Service function chain placement
Machine learning
Neural-networks
Multi-access edge computing

ABSTRACT

5G mobile network technology promises to deliver unprecedented ultra-low latency and high data rates, paving the way for many novel applications and services. *Network Function Virtualization* (NFV) and *Multi-access Edge Computing* (MEC) are two of the technologies that are expected to play a pivotal role in 5G to achieve ambitious Quality of Service requirements of such applications. While NFV provides flexibility by enabling network functions to be dynamically deployed and inter-connected to realize Service Function Chains (SFC), MEC brings the computing capability to the edges of the mobile network thus reducing latency and alleviating the transport network load. However, adequate mechanisms are needed to meet the dynamically changing network service demands, to optimally utilize the network resources while, at the same time, making sure that the end-to-end latency requirement of services is always satisfied.

In this work, we first propose machine learning models, in particular neural-networks, that can perform auto-scaling by predicting the required number of virtual network function instances based on the traffic demand, using the traffic traces collected over a real-operator commercial network. We then employ Integer Linear Programming (ILP) techniques to formulate and solve a joint user association and SFC placement problem, where each SFC represents a service requested by a user with end-to-end latency and data rate requirements. Finally, we propose a heuristic to address the scalability concern of the ILP model.

1. Introduction

The 5*th* generation of mobile networks is expected to support high data rates, extremely low-latency, high reliability, the capability to extend access to distributed computation and storage facilities in addition to connectivity and bandwidth [1]. These characteristics of the 5G systems open the door for many novel Ultra-reliable low-latency (URLLC) applications such as augmented/virtual reality and autonomous driving, whose ambitious Quality of Service (QoS) requirements cannot be satisfied by the preprocessors of the 5G networks. Therefore, the 5G architecture needs to incorporate new technologies such as Multi-access Edge Computing (MEC) [2] and Network Function Virtualization (NFV) [3], to meet the insatiable data rate and low-latency requirements of the applications mentioned above [4].

The basic idea of MEC is to bring computing capabilities and applications closer to the end-users, from cloud data centers to the edges of the cellular network, therefore, reducing the delay experienced by the users and alleviating the transport network load. Consequently, the ETSI MEC Industry Specification Group proposes three possible MEC deployment options in 5G networks, collocated with the gNodeB (*gnb.mec*) or collocated with an aggregation point (*ap.mec*) or collocated with the 5G core network (5*gc.mec*) [5], as shown in Fig. 1. The closer the MEC nodes are towards the end-users, the scarcer their computational resources become. It is important to mention that the integration of MEC into 5G network requires the collocation of a User Plane Function (UPF) element with each MEC node to reap the benefits of the MEC system [5]. The functionality of the UPF in MEC systems is illustrated in Section 2, together with other relevant concepts and terminologies in 5G mobile network.

NFV, on the other hand, decouples network functions (e.g., UPF) or MEC applications from their dedicated proprietary hardware and deploys them as virtualized software entities on commodity servers [6]. In our work, we use the generic term VxF to refer to either UPF virtualized network function (VNF) or virtualized MEC

[☆] This work has been performed in the framework of the European Unions Horizon 2020 project 5G-CARMEN co-funded by the EU under grant agreement No 825012. The views expressed are those of the authors and do not necessarily represent the project. The Commission is not liable for any use that may be made of any of the information contained therein.

[*] Corresponding author.

E-mail addresses: t.subramanya@fbk.eu (T. Subramanya), d.harutyunyan@fbk.eu (D. Harutyunyan), rriggio@fbk.eu (R. Riggio).

https://doi.org/10.1016/j.comnet.2019.106980
1389-1286/© 2019 Elsevier B.V. All rights reserved.

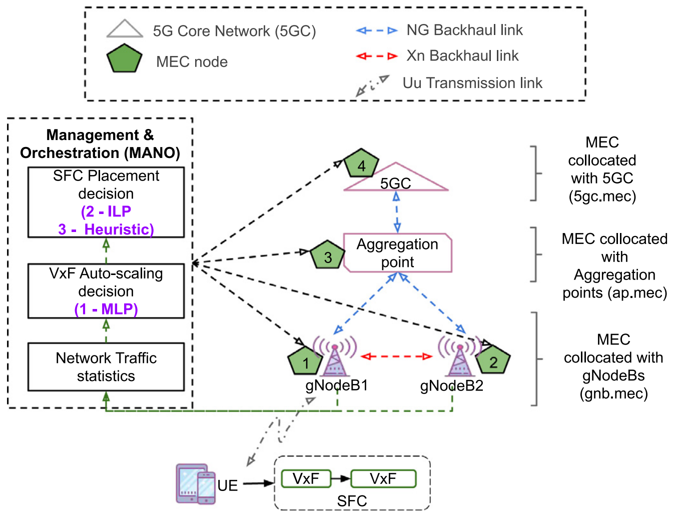
5G Core Network (5GC)
MEC node
NG Backhaul link
Xn Backhaul link
Uu Transmission link
Management & Orchestration (MANO)
SFC Placement decision
(2 - ILP
3 - Heuristic)
VxF Auto-scaling decision
(1 - MLP)
Network Traffic statistics
5GC
Aggregation point
gNodeB1
gNodeB2
MEC collocated with 5GC (5gc.mec)
MEC collocated with Aggregation points (ap.mec)
MEC collocated with gNodeBs (gnb.mec)
UE
VxF
VxF
SFC

Fig. 1. An example of a distributed MEC-NFV System Architecture.

application function (VMAF). VxFs require a specific computational capacity (e.g., CPU) to be instantiated and can be chained together, forming a Service Function Chain (SFC) that represents a specific service, with a guaranteed latency and data rate requirements, that can be requested by User Equipments (UEs). As shown in Fig. 1, there are multiple locations (e.g., gnb.mec, ap.mec, 5gc.mec) for instantiating VxFs, which can be shared by many UEs.

In the described network scenario, given the UEs with their SFC demands, the natural question that arises is how to associate UEs, place their SFCs, allocate sufficient VxF instances and resources to make sure that the UEs SFC requirements are satisfied while the network resources are used efficiently? Moreover, the mobility patterns of UEs result in non-uniform traffic distribution within the mobile network [7]. Consequently, the number of VxF instances required to manage load variations and to meet performance guarantees is expected to fluctuate frequently. Towards this end, auto-scaling of VxFs, in addition to UEs association and their SFC placement, is thought to be an essential requirement for successful management and orchestration (MANO) of resources and services in 5G networks.

Most of the existing literature address either the problem of VxF autoscaling [8] or the placement of SFC in distributed MEC nodes [9]. In this paper, we first advocate that VxFs of SFC has to be proactively scaled in synergy with varying network traffic dynamics to avoid service disruption. Based on those scaling decisions, the VxFs need to be dynamically placed in distributed MEC nodes, to minimize end-to-end latency and to meet data rate requirements. To the best of our knowledge, we are the first to address the combined challenges in VxF auto-scaling and placement of SFCs within a distributed MEC-NFV environment, based on the real-operator mobile network traces. This work has three main contributions (also as depicted in Fig. 1) compared to our previous work [10]:

(i) Our previous work just considered a neural-network-based Multi-layer Perceptron (MLP) *classifier* model to estimate the required number of UPF instances as a function of the network traffic they should process in each base station. In this extended work, we also consider the neural-network-based MLP *regressor* model to perform the same objective, and we analyze the performance of both models. Additionally, our previous work only examined the QoS-prioritized neural-network model while in this work, we also examine the cost-prioritized neural-network model and compare the performance of both methods. The output from the best performing MLP model i.e., 'the number of UPF instances' is fed as an input to the Integer Linear Programming (ILP) model. It is to be noted that the 5G mobile network dataset employed in our model is obtained by a commercial operator in Armenia.

(ii) Our previous work used the ILP technique to solve solely the VxF (not SFC) placement problem with specific latency demands as requested by UEs. However, the delay values used in our model were static values based on other similar works, and also the model did not consider UE associations to gNodeBs and their requested data rates. In this extended work, we employ ILP technique to formulate and solve a *'joint UE association and SFC placement problem'*, where each SFC is composed of several VxFs interconnected through virtual links, with specific latency and data rate demands as requested by UEs positioned in diverse areas of the 5G mobile network. Furthermore, we also develop a comprehensive end-to-end latency model considering radio delay (comprised of UE processing delay, over-the-air transmission delay, gNodeB processing delay, scheduler queuing delay, and Hybrid automatic repeat request (HARQ) retransmission delay), backhaul network delay (comprised of propagation delay and transmission delay both in Xn and NG interfaces as shown in Fig. 1), and SFC processing delay for 5G mobile networks.

(iii) Our previous work did not propose any heuristic algorithm since the ILP model was simplistic while in this work, we propose a heuristic algorithm with the same objective as that of ILP to address the scalability problem of ILP.

The rest of this paper is organized as follows. Sections 2 and 3 describes the necessary background and related work, respectively. Section 4 describes the proposed MLP classifier and MLP regressor models and evaluates their performance. In Section 5, we model latency-optimal SFC placement problem whileSection 6 formulates the problem using ILP and also proposes a heuristic algorithm. InSection 7, we perform several experiments to evaluate our proposed SFC placement solutions. Finally, we conclude the paper in Section 8.

2. Background: 5G mobile network

In this section, we briefly introduce the 5G mobile network architecture (see Fig. 2) and provide an overview of the basic concepts and terminologies in 5G. At a very high level, a 5G mobile network is composed of two major elements: the 5G Access Network and the 5G Core Network (5GC). The 5G Access Network comprises one logical node, the next-generation NodeB (gNodeB), which connects to the User Equipments (UEs), providing control plane and user plane services. The gNodeBs are interconnected to each other using the Xn interface. The 5GC consists of many logical nodes such as Access and Mobility Management Function (AMF), Session Management Function (SMF), and User Plane Function (UPF). The gNodeBs are connected to the 5G Core Network

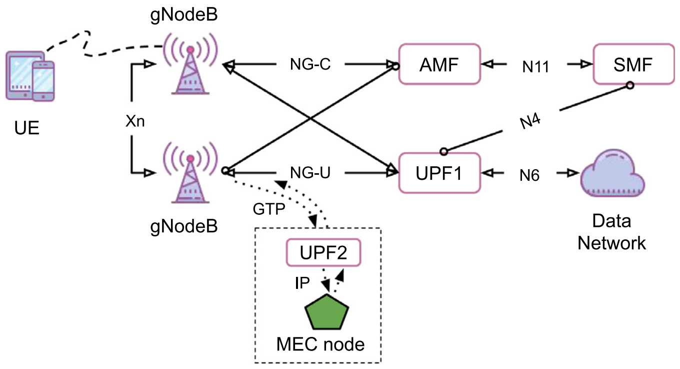

Fig. 2. 5G mobile network overview.

through NG interfaces, more specifically to the AMF through NG-C control plane interface and to the UPF1 through NG-U user plane interface. The components within the 5GC are also interconnected using standardized interfaces.

The scheduler entity in the gNodeB is responsible for deciding which UEs should be allocated air interface resources, i.e., Physical Resource Blocks (PRBs) on Transmission Time Interval (TTI) (e.g., $1ms$ or $0.5ms$ or $0.25ms$ or $0.125ms$) basis and how much PRBs should be allocated to send or receive data. Once a UE attaches to the network using control plane signaling (e.g., attach process), it can send/receive data to/from the Packet Data Networks (PDN) using the GPRS Tunneling Protocol (GTP). The uplink UE traffic received by the gNodeB over its air interface is encapsulated into a GTP packet and then delivered to the UPF1 over the NG-U interface. This GTP tunnel is terminated at the UPF1 which removes the GTP header and forwards the UE traffic to its intended destination (e.g., the Internet).

Besides, consider a MEC node being placed in between the gNodeB and the 5GC to support low-latency applications. For the MEC applications to operate, it needs to have access to the UE IP traffic. Therefore, the integration of MEC in the 5G network requires the colocation of a UPF network function with the MEC node. The UPF takes care of performing a stateful termination and recreation of the GTP session concerning the UE. As we see in Fig. 2, the GTP-encapsulated UE traffic is redirected from the gNodeB to the UPF2. Here the GTP tunnel is terminated, and the UE IP traffic, now accessible, is redirected to the MEC Node running MEC applications. On the way back, the GTP tunnel is recreated by UPF2, and the response is delivered back to the UE.

3. State of the art

ETSI NFV Industry Specification Group defines network service as a composition of one or more VNFs that are chained together. Each VNF requires a specific amount of resource to process the traffic flowing through it. To deploy a network service, the operator needs to find the right placement of VNFs complying with various resource constraints and service latency agreements. Once the hosts are selected and the VNFs deployed, resource requirements for the VNFs may vary due to traffic fluctuations. To meet these demands, a resource allocation algorithm is needed that can automatically allocate/release resources to a VNF (vertical scaling) or add/remove one or more VNF instances (horizontal scaling).

3.1. Virtual network function auto-scaling.

Previous works on VNF auto-scaling can be divided into two categories: reactive mode and proactive mode.

In reactive mode, threshold levels can be either statically predefined or dynamically updated. In [11–13], the authors propose scalability mechanisms based on static thresholds. They define two threshold levels ($scalein_{thr}$ and $scaleout_{thr}$) to determine if the load reduces below or exceeds above the respective limits and accordingly trigger the scaling process. However, such techniques may result in an oscillating behaviour affecting the overall system performance. On the other hand, [14,15] propose mechanisms such as queuing theory and reinforcement learning, which allows the scaling policy to be improved based on dynamic or adaptive thresholds. Although it performs better than static approaches, it remains a reactive solution with similar weaknesses.

In proactive mode, forecasting techniques (e.g., machine learning) are applied to allow the systems to automatically learn and to anticipate future needs, based on which scalability decisions are taken. For example, the authors in [16] propose a solution to forecast CPU usage based on a historical dataset using time series model. Other authors such as Mijumbi et al. [17] and Mestres et al. [18] addresses the problem of managing VNF resource fluctuations by predicting resource requirements using ML techniques and thereby enhancing the performance of the resource allocation algorithm.

In contrast to these works which targets data centers, our approach investigates the problem of proactive auto-scaling in a distributed MEC-NFV deployment. Moreover, we use real-operator traffic traces to generate training sets required for predicting auto-scaling decisions, unlike other works that are based on simulated datasets.

3.2. Service function chain placement.

There already exists some literature on the SFC placement problem with certain end-to-end latency needs that need to be satisfied [9], [19], and [20]. In [9], the authors present a delay-aware SFC placement problem such that VxFs forming SFCs are placed so as to satisfy end-to-end latency demands while utilizing network resources in an effective manner. A joint VxF placement and CPU allocation problem is studied in [19] and an optimization problem is formulated by employing a queuing-based model to minimize the ratio between the actual and the maximum allowed latency, for all SFC requests. The authors in [20] study the problem of VxF instantiation and migration with a goal of minimizing SFC delays. However, all these studies do not consider UE processing time, gNodeB processing time, and propagation or transmission time over the air interface. Besides, none of the above studies consider heterogeneous MEC nodes, which increases the search space causing the SFC placement problem to grow cumbersome.

In contrast to the above SFC placement solutions, we consider the joint problem of user association and SFC placement which allows the optimization of end-to-end latency according to user locations, SFC latency and data rate requirements, and computing/networking resource availabilities. Furthermore, our proposed latency model stands out from the existing delay models within the context of 5G mobile network.

4. Machine learning-driven proactive *'UPF' auto-scaling*

In this section, we create two types of neural-network based MLP models, a *classifier* and a *regressor*, that can identify and exploit hidden patterns in network traffic load instances to predict UPF scaling decisions ahead of time. In particular, we illustrate on different steps involved in creating our models and eventually evaluate them based on several performance metrics [21].

4.1. Problem description

We investigate how to map traffic load statistics X to VNF scaling decisions Y using supervised learning, which involves learning from a training set of data. The traffic load statistics X include measurements from a commercial operator 5G mobile network. The VNF scaling decisions Y refer to the required number of UPFs to process incoming traffic with an objective to either maximize QoS or minimize cost. The details on the composition of X and Y are discussed in Section 4.4.

The X and Y metrics evolve over time, influenced mainly by the mobile network traffic dynamics and the actual number of mobile users. Consequently, the combined evolution of X and Y metrics is modeled as a time series $\{(x_t, y_t)\}$. Our goal is to determine the distribution of scaling decision metric Y constrained on knowing the traffic load metric $x \in X$.

Employing the statistical learning framework, X and Y are modeled as random variables. We assume that each sample (x_t, y_t) in the training set is obtained from the conditional probability distribution of (X, Y). Further, we suppose that x_t is multi-dimensional (multi-variate) and y_t is one-dimensional (uni-variate). In this formalism, the inference problem consists of finding a model F: $x \to P(Y|x)$ for $x \in X$, so as to maximize the likelihood function $L(\{P(y_t|x_t)\})$, which can be attained by minimizing the loss/error function.

In this work, a neural-network called Multilayer Perceptron (MLP) is used to estimate the parameters of the model to predict the probability distribution $P(Y|x)$. We select artificial neural-network in our approach for three reasons:

(i) it has proven its potential in identifying traffic patterns due to its effectiveness in predicting time-series problems, whether periodic or not [22].

(ii) it can build new customized features through hidden layers and fit nonlinear activation functions when a specific mathematical definition is not available.

(iii) it can represent both linear, piecewise-linear and non-linear relationships and learn these relationships directly from the data.

4.2. Multilayer perceptron (MLP)

An MLP is a class of feed-forward artificial neural network, consisting of at least three layers of nodes (neurons): an input layer, one or more hidden layers, and an output layer, as shown in Figs. 3 and 4. These nodes are fully interconnected in the form of a directed graph, starting from the input to the output. All nodes except the input nodes have an associated activation function, which is used to compute the node output based on the weighted inputs from other nodes. An MLP model is trained through a back-propagation mechanism using gradient-descent as an optimization algorithm, where the weights between the nodes are adjusted iteratively for minimizing the error function.

MLP Classifier. In classification, a relu activation function is used for all hidden layer nodes, and a softmax activation function is used for the output layer nodes. The output is a vector containing the probabilities that sample $x \in X$ belongs to each class, which is equivalent to a categorical probability distribution (as seen in Fig. 3). The final result is the class with the highest probability. With a categorical cross-entropy loss function, the network parameters are chosen to minimize the following:

$$E = -\sum_{l=1}^{C} b_{x,l} \log(p_{x,l}) \quad (1)$$

where C is the number of classes, b is the binary indicator (0 or 1) whether class label l is the correct classification for input x, and p is the predicted probability that input x belongs to class l. Here, a separate loss is calculated for each class label per input, and the result is the sum of all those losses.

MLP Regressor. In regression, a relu activation function is used for all hidden layer nodes, and a linear activation function is used for the output layer nodes. The output is a real-valued quantity predicted based on the input sample $x \in X$ (as seen in Fig. 4). With a mean-squared-error (MSE) loss function, the network parameters are chosen to minimize the following:

$$MSE = 1/n \sum_{i=1}^{n} (Y_a - Y_p)^2 \quad (2)$$

where n is a vector of predictions generated from a sample of n data points while Y_a and Y_p are the actual and predicted values of the samples.

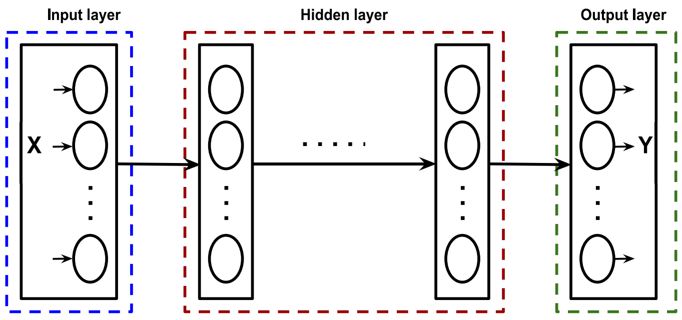

Fig. 3. Structure of the MLP classifier model.

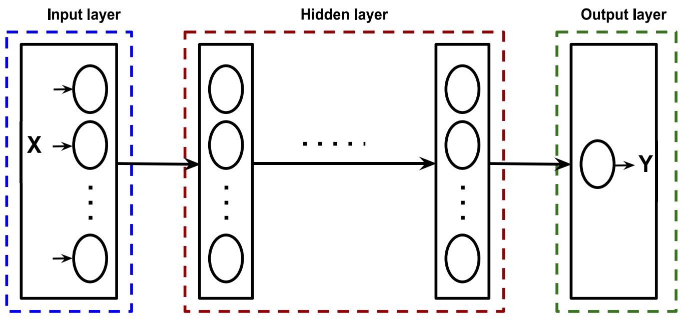

Fig. 4. Structure of the MLP regressor model.

4.3. Modeling MLP in keras

Keras is an open-source neural-network Python library capable of running on top of Theano [23] or TensorFlow [24]. It is characterized by a clean, uniform, and streamlined high-level API, allowing users to rapidly define, train, and evaluate neural network models [25].

In Keras, the structure of the neural network model can be defined in a modular way, as a sequence of standalone and fully configurable modules, which can be readily plugged together. Keras offers several predefined neural layers such as a dense layer, a recurrent layer, and a convolutional layer. A wide range of activation functions is also available including relu, sigmoid, softmax, tanh, to name a few. Similarly, many predefined loss functions (e.g., mean squared error, cross entropy) and regularization schemes (e.g., dropout) are supported. Also, since Keras performs backpropagation automatically, users do not need to implement it. Moreover, numerous approaches are available to partition the dataset into training, validation, and test sets.

To implement an MLP in Keras, we construct a sequential model with a number of predefined dense layers and their corresponding activation functions. We then configure the learning process of the model by choosing an optimizer, a loss function (Eq. (1) or Eq. (2)), and a list of metrics to be reported. Lastly, the model is trained with an objective to minimize the loss function and then evaluated.

4.4. Collecting data and feature engineering

The different steps that we followed in creating our MLP models are as follows:

4.4.1. Data collection

The dataset utilized in this work is generated from a commercial operator by monitoring the mobile network traffic load on 6 base stations, with each base station having 10 cells, for a period of 8 consecutive days. The traces in the dataset are in the form of a time series $\{(x_t, y_t)\}$ and we interpret this time series as a set of samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. The traces are collected on an hourly timescale.

Table 1
Default set of features available in the dataset.

Default features ($X_{default}$)
1. gNodeB ID.
2. Date.
3. Time-stamp t.
4. Average number of users between t and $t-1$ in each cell.
5. Maximum number of users between t and $t-1$ in each cell.
6. Average downlink user throughput in each cell.
7. Average uplink user throughput in each cell.
8. Traffic load measured in each cell at time t, given by $\lambda(t)$.

Table 2
Constructed set of features from the dataset.

Constructed features ($X_{constructed}$)
9. Traffic load measured in each cell at time $t-1$, given by $\lambda(t-1)$.
10. Traffic load measured in each cell at time $t-2$, given by $\lambda(t-2)$.
11. Traffic load measured in each cell at time $t-3$, given by $\lambda(t-3)$.
12. Traffic load measured in each cell at time $t-4$, given by $\lambda(t-4)$.
13. Change in traffic load in each cell from time t to $t-1$.
14. Change in traffic load in each cell from time $t-1$ to $t-2$.
15. Change in traffic load in each cell from time $t-2$ to $t-3$.
16. Change in traffic load in each cell from time $t-3$ to $t-4$.
17. Weekday or weekend.

4.4.2. Feature extraction

We now describe the input feature sets $X_{default}$ and $X_{constructed}$, which, when combined, is referred to as X, as well as the output classes or real-valued quantities Y.

The $X_{default}$ feature set includes 8 numeric features that are already available in the dataset, as described in Table 1. In addition to these default features, we construct 9 numeric features ($X_{constructed}$) from the basic dataset, as shown in Table 2, using a process called *feature transformation* by extending backward from time t. These constructed features contain information or patterns on how the traffic load evolves, therefore assisting in proactive VNF scaling decisions.

4.4.3. Definition of classes or real-valued quantity 'Y'

The next step is to define how we generate output classes or real-valued quantity Y, which the MLP classifier or regressor tries to predict, respectively. In VNF autoscaling, there is a tradeoff between QoS and cost. More UPF instances (i.e., resources) need to be allocated to guarantee QoS, but allocating more resources raises the cost. Therefore, we propose two different approaches: (i) QoS favored classifier/regressor (Q-classifier/Q-regressor) and Cost favored classifier/regressor (C-classifier/C-regressor).

In Q-classifier/Q-regressor, the network operator gives priority to QoS over the cost. The autoscaling decision at step n considers future traffic demands until the next autoscaling step $n+1$. Therefore, the class value or the target variable value is generated as follows:

$$Y_Q = min\left(vnf_{max}, max\left(\frac{\lambda(t)}{\gamma}\right)\right) \forall t \in \{\tau(n), \ldots\ldots, \tau(n+1)\} \tag{3}$$

where t are the timestamps containing traffic data samples between steps n and $n+1$ (including $\tau(n)$ and $\tau(n+1)$), $\lambda(t)$ is the traffic load in a cell at time t, γ is the maximum traffic load a single UPF can handle, and vnf_{max} is the maximum number of UPFs per cell that can be hosted on the MEC node.

In C-classifier/C-regressor, the network operator chooses to neglect short-lived bursty traffic between steps n and $n+1$ to avoid over-provisioning of UPFs, therefore minimizing cost and enduring short-lived degradations. Consequently, the autoscaling decision considers measured traffic load only at step n and at next auto-scaling step $n+1$. Therefore, the class value or the target variable value is generated as follows:

$$Y_C = min\left(vnf_{max}, max\left(\frac{\lambda(\tau(n))}{\gamma}, \frac{\lambda(\tau(n+1))}{\gamma}\right)\right) \tag{4}$$

where $\tau(n)$ is the time at which step n occurs and $\tau(n+1)$ is the time at which step $n+1$ occurs.

It is to be noted that in our work, we examine the performance metrics for Q-classifier, Q-regressor, C-classifier, and C-regressor for two cases: (i) autoscaling decisions performed on *one* hour time-intervals and (ii) autoscaling decisions performed on *two* hour time-intervals. Although our traffic traces are collected on hourly time intervals, our model is generic enough to handle lower time interval granularities (e.g., 5-min time interval data samples) and different auto-scaling steps (e.g., 1 hour, 5 hours).

4.4.4. Feature subset selection

Next, we identify the dominant features from our feature list based on their influence on classification 'accuracy' or regression 'R-squared' values using Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA) techniques.

RFE is a greedy optimization technique that strives to find the best performing feature subset. It repeatedly generates models and keeps aside the best or the worst performing feature in each iteration. The next model is constructed with the remaining features until all the features are depleted. It then ranks the features based on the order of their elimination. With MLP, it is difficult to understand which input features are relevant and which are not. The reason being, each input feature has multiple coefficients that are linked to it - each corresponding to one node of the first hidden layer. Additional hidden layers make it even more challenging to decide how big of an impact the input feature has on the final prediction. Therefore, we apply the RFE technique on a linear support vector machine model to find the optimal number of features and use them in creating our MLP models. After ranking, features 8, 9, 10, 11, and 12 are ranked highest, which implies that measured loads closer to the scaling decision time are the crucial features. Features 2, 17, and 3 are ranked 2nd, 3rd, and 4th, respectively. The rest of the features are ranked in the following order: 14, 15, 13, and 16. Finally, features 1, 4, 5, 6, and 7 are recommended not to be used in the model (RFE returns 'false').

We have also validated this observation using Principal Component Analysis (PCA), a statistical method to find correlated features and their impact on classification and regression. In PCA, the first principal component has the most notable variance, accounting for much of the variability in the data samples. Our PCA lists a combination of features 8, 9, 10, 11, and 12 as the first principal component, indicating similar conclusions as RFE.

Based on the ranking of these features, we use only 12 features (eliminating 1, 4, 5, 6 and 7 from Table 1) that provides the best results for our MLP models.

4.4.5. Dataset decomposition

Once data is collected and features extracted/selected, the dataset is decomposed into training, validation and test datasets. We use a rule-of-thumb decomposition conforming to 60%/20%/20% between the training, validation and test datasets, respectively. The data samples chosen for training and validation (i.e., close to 6 days of data from 6 gNodeBs) are the most balanced data in our dataset compared to samples from other days that were used for testing (close to 2 days). Even then, the training samples are slightly imbalanced in classes/target real-valued quantities which might result in overfitting the model (i.e., a condition where a statistical model begins to output a random class or error value outside the original dataset), and therefore we look at other performance metrics such as confusion matrix, precision, recall, and F1-score for MLP classifier and R-squared value for MLP regressor, which can provide more insight into the performance of our MLP models than traditional classification accuracy and regression mean squared error, which are excellent measures only if the datasets are entirely symmetric. Finally, if the performance metrics indicate overfitting, the K fold cross-validation technique can be used to generate multiple mini train-test splits to tune our MLP models, which was not necessary in our case.

4.5. Classification and regression using neual networks

Finding the parameters of a neural-network model means searching for the best hyper-parameters of the MLP that can make the best predictions on the input. We applied *grid search* and *baby-sitting* as search strategies to perform an extensive search on the space of hyper-parameters to find the most accurate neural-network classifier and regressor. This process included finding the number of hidden layers and nodes, the batch size, the regularization parameter, the learning rate of the optimizer, and the number of epochs. We encountered the process of finding hyper-parameters time-consuming and hard, which assures that this topic still requires significant research. Our search space for finding optimal hyperparameters for MLP models are as follows:

- Hidden layers: 1 to 5.
- Nodes in each hidden layer: 12 to 30 in intervals of 3.
- Optimizer: adam, SGD, RMSprop.
- Learning rate: 0.1, 0.01, 0.001.
- Batch size: 100 to 500 in intervals of 100.
- Number of epochs: 100 to 500 in intervals of 100.

We eventually found the architecture of the neural network that performs best on our traffic load traces and is described as follows. The structure includes one input layer with 12 nodes (i.e., one for each input feature), three hidden layers with 12, 24 and 12 nodes, respectively, and an output layer with 10 nodes for MLP classifier (i.e., one for each output class) and 1 node for MLP regressor. The regularization parameter used is 0.01, the optimizer is based on stochastic gradient approach with a constant learning rate of 0.001, the batch size is fixed to 100, and the number of epochs equals 300.

4.6. MLP Model evaluation

We consider that MEC nodes in proximity to the gNodeBs are capable of hosting UPFs on their NFV infrastructure. We assume the link bandwidth capacity to be 20 Gbps and each VNF can process a maximum of 200 Mbps traffic without QoS degradation. We consider horizontal VNF auto-scaling with each MEC node capable of hosting $100 (20Gbps/200Mbps)$ VNFs and $vnf_{\max} = 10$, i.e., a maximum of 10 VNFs can be hosted per cell. These assumptions are derived based on the evaluations performed by authors in [26]. If traffic load increases, additional VNF instances are deployed to meet QoS/cost requirements, whereas if traffic load decreases, VNF instances are removed to save operational expenses.

MLP Classifier. Once the MLP classifier models are created as discussed before, a test dataset is used to assess the performance of the model in predicting outcomes. The test outcomes can be classified into four groups: True Positive (TP) and True Negative (TN) are when the model correctly predicts actual positive and negative instances, respectively. Whereas, False Positive (FP) and False Negative (FN) are when the model makes incorrect predictions for negative and positive actual instances, respectively. Therefore, we consider four performance metrics to evaluate our MLP classifier model: accuracy, precision, recall, and f-measure, as given by Eqs. (5)–(8), respectively.

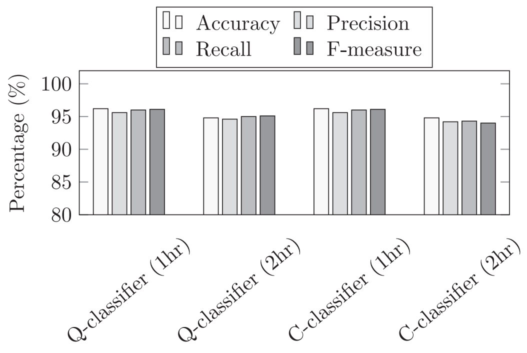

Fig. 5. Comparision of the proposed MLP classifier models for VNF auto-scaling.

$$Accuracy = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{5}$$

$$Precision = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i}{TP_i + FP_i} \tag{6}$$

$$Recall = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i}{TP_i + FN_i} \tag{7}$$

$$F_{measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

where C is the number of classes in the MLP model.

Accuracy is the most intuitive performance measure that gives the proportion of true predictions among the total number of predictions observed. However, accuracy is an excellent measure only if the datasets are entirely symmetric, i.e., false positives and false negatives are almost the same. Therefore, other performance metrics need to be considered when evaluating a model. *Precision* is a measure of correctly predicted positive observations to the total predicted positive observations. It is a good measure to determine when the cost of FP is high. In the case of VNF auto-scaling, a high number of FPs results in over-provisioning of resources leading to increased operational costs. On the other hand, *Recall* is a measure that calculates how many of the actual positives are captured in our model by labeling it as positive. It is a good measure to determine when the cost of FN is high. In the case of VNF auto-scaling, a high number of FNs results in under-provisioning of resources leading to QoS degradation. Finally, *F-measure* is the weighted average of precision and recall, and it is used when there is an uneven class distribution.

Fig. 5 compares the performance of four proposed MLP classifier models: Q-classifier with scaling decisions every hour, Q-classifier with scaling decisions every two hours, C-classifier with scaling decisions every hour, and C-classifier with scaling decisions every two hours. We use 6912 samples for training, 2304 samples for validation, and 2304 samples for testing. The Q-classifier/C-classifier with scaling decisions taken every hour outperforms other two models where scaling decisions are taken every two hours in all measures with 96.2% accuracy, 95.6% precision, 96% recall, and 96.2% f-measure.

Tables 3–5 reports the confusion matrix concerning the test data samples for Q-classifier/C-classifier models with scaling decisions every hour, Q-classifier model with scaling decisions every two hours, and C-classifier model with scaling decisions every two hours, respectively. It gives a breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned). For example, if we observe Table 3 on the 2nd row, on 5

Table 3
Confusion matrix for the Q-classifier (1hr)/ C-classifier (1hr) models.

Class	1	2	3	4	5	6	7	8	9	10
1	685	9	0	0	0	0	0	0	0	0
2	5	384	6	0	0	0	0	0	0	0
3	0	6	404	7	0	0	0	0	0	0
4	0	0	7	280	6	0	0	0	0	0
5	0	0	0	3	206	8	0	0	0	0
6	0	0	0	0	9	101	3	0	0	0
7	0	0	0	0	0	2	70	5	0	0
8	0	0	0	0	0	0	4	43	3	0
9	0	0	0	0	0	0	0	3	30	0
10	0	0	0	0	0	0	0	0	1	14

Table 4
Confusion matrix for the Q-classifier (2hr) model.

Class	1	2	3	4	5	6	7	8	9	10
1	322	6	0	0	0	0	0	0	0	0
2	2	165	4	1	0	0	0	0	0	0
3	0	3	206	5	0	0	0	0	0	0
4	0	0	2	142	6	0	0	0	0	0
5	0	0	0	2	112	6	1	0	0	0
6	0	0	0	0	6	53	1	0	0	0
7	0	0	0	0	0	3	41	4	0	0
8	0	0	0	0	0	0	3	24	1	0
9	0	0	0	0	0	0	0	3	19	0
10	0	0	0	0	0	0	0	0	0	9

Table 5
Confusion matrix for the C-classifier (2hr) model.

Class	1	2	3	4	5	6	7	8	9	10
1	320	8	0	0	0	0	0	0	0	0
2	3	165	2	2	0	0	0	0	0	0
3	0	6	204	4	0	0	0	0	0	0
4	0	0	5	140	5	0	0	0	0	0
5	0	0	2	6	106	7	0	0	0	0
6	0	0	0	0	6	53	1	0	0	0
7	0	0	0	0	0	2	42	3	1	0
8	0	0	0	0	0	0	4	22	2	0
9	0	0	0	0	0	0	0	4	17	1
10	0	0	0	0	0	0	0	0	1	8

instances class 2 is misclassified as class 1, and on 6 instances class 2 is misclassified as class 3. Similarly, if we observe Table 4 on the 2nd row, on 2 instances class 2 is misclassified as class 1, and on 4 instances class 2 is misclassified as class 3, and on 1 instance class 2 is misclassified as class 4.

MLP Regressor. Once the MLP regressor models are created as discussed before, a test dataset is used to assess the performance of the model in predicting outcomes. We implement four custom performance metrics in Keras to evaluate our MLP regressor models: mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R^2-score, as given by Eqs. (9)–(12).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_a - Y_p| \tag{9}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_a - Y_p)^2 \tag{10}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_a - Y_p)^2} \tag{11}$$

$$R^2_{score} = 1 - \frac{\sum_{(Y_a-Y_p)^2}}{\sum_{(Y_a-Y_m)^2} } \tag{12}$$

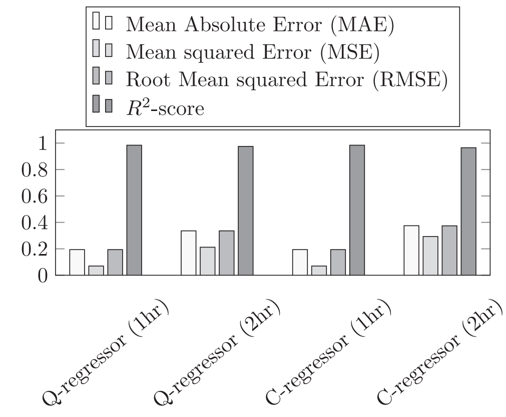

Fig. 6. Comparision of the proposed MLP regressor models for VNF auto-scaling.

where n is the number of test data samples, Y_a is the actual value of Y, Y_p is the predicted value of Y, and Y_m is the mean value of Y.

MAE estimates the average magnitude of errors in a set of forecasts, without considering their direction (i.e., the average of the absolute values of differences between the forecast and the corresponding observation).*MSE* measures the average of the squares of the errors (i.e., the average squared difference between the estimated values and the actual values). *RMSE* is a quadratic scoring rule which measures the average magnitude of the error (i.e., the difference between the estimated values and the actual values are each squared and then averaged over the sample). Then, the square root of the average is estimated. Considering the errors are squared before they are averaged, the RMSE adds a relatively high weight to big errors. Therefore, RMSE is most useful when large errors are undesirable. The RMSE is always larger or equal to the MAE, the greater difference between them, the higher the variance in the individual errors in the sample. If the RMSE is equal to the MAE, then all the errors are of the same magnitude. R^2*-score* (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 are interpreted as percentages. The higher the value is, the better the model is. It is equivalent to the accuracy metric in classification problems.

Fig. 6 compares the performance of four proposed MLP regressor models: Q-classifier with scaling decisions every hour, Q-classifier with scaling decisions every two hours, C-classifier with scaling decisions every hour, and C-classifier with scaling decisions every two hours. We use 6912 samples for training, 2304 samples for validation, and 2304 samples for testing. The Q-regressor/C-regressor (1hr) performs the best among the four models in all measures with 0.194 MAE, 0.0696 MSE, 0.194 RMSE and 98.43% R^2-score/accuracy. Moreover, considering that the MAE and the RMSE values are equal to each other, it is safe to say that the model has no large errors in the required number of UPF predictions.

MLP Classifier vs MLP Regressor. Fig. 7 shows the prediction results of VNF auto-scaling (for a full day) on test dataset for most reliable MLP classifier and MLP regressor models based on the metrics mentioned earlier, where we display the prediction performance on all six MEC nodes, aggregated over all 10 cells for each gNodeB. In the figure, the blue line represents the actual output generated from the dataset, the red line means the predicted VNF scaling decisions using Q-classifier/C-classifier (1hr) models, and the brown line represents the predicted VNF scaling decisions using Q-regressor/C-regressor (1hr) models. As we can observe, both classifier and regressor models introduced in this study can accurately follow the pattern of actual data, which point out the strong predicting capability of our models. However, Q-regressor/C-regressor (1hr) models (accuracy of 96.2%) perform slightly better

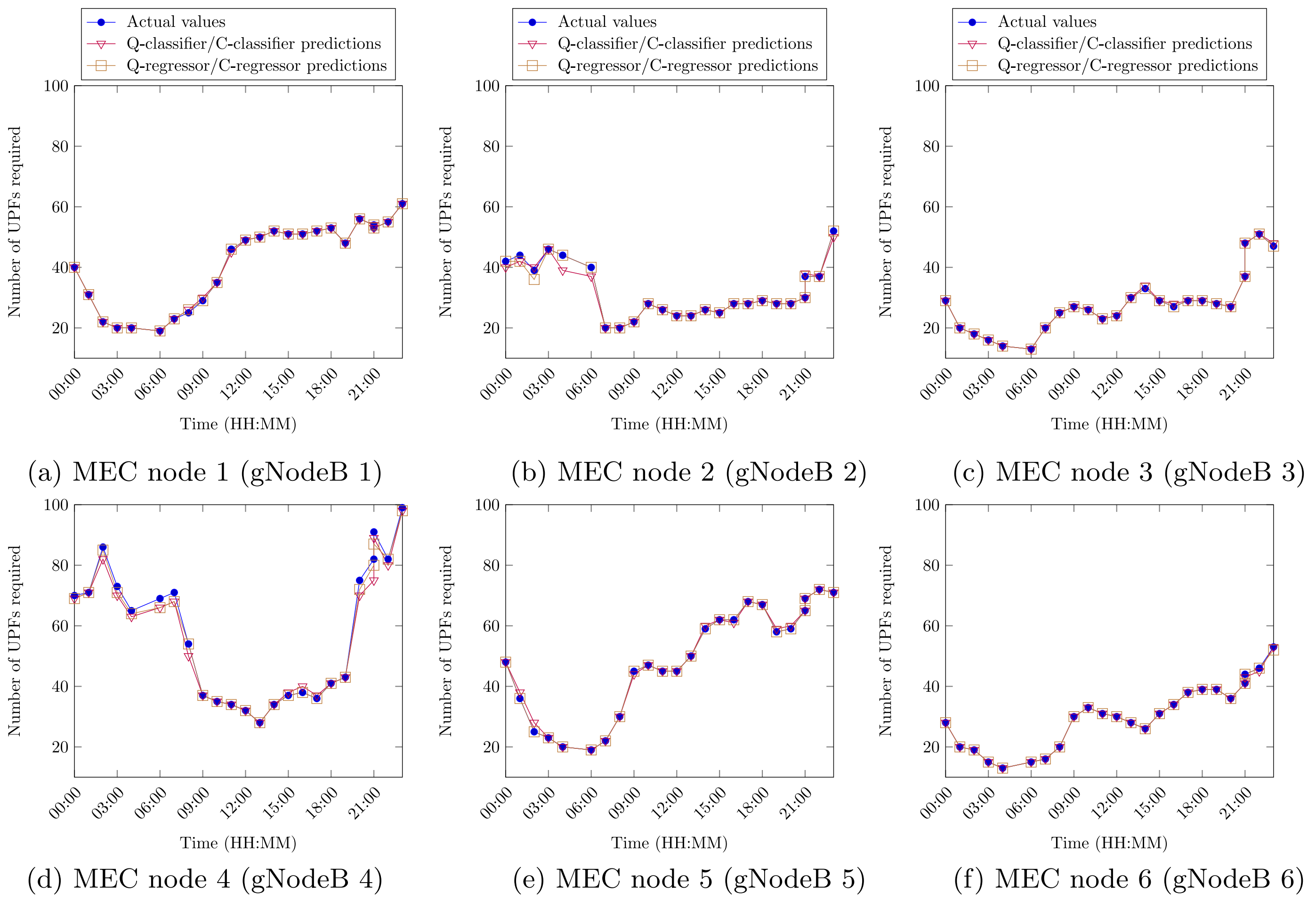

Fig. 7. Prediction results on the number of UPFs required at each MEC node based on the proposed MLP models.

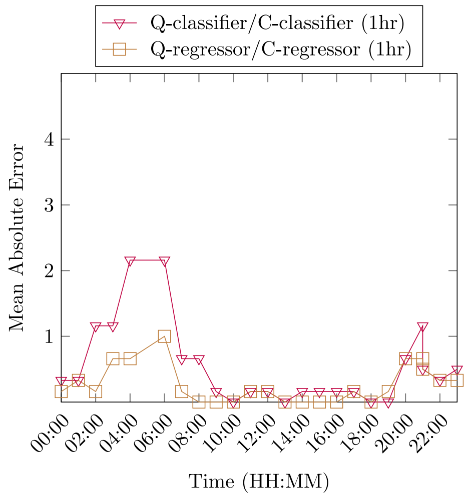

Fig. 8. Comparison of Mean Absolute Error between the best performing MLP classifier and MLP regressor models.

than Q-classifier/C-classifier (1hr) models (accuracy of 98.43% or R^2-score of 0.984).

Fig. 8 depicts the MAE between the actual and predicted values of VNF auto-scaling decisions on test dataset for best performing MLP classifier and MLP regressor models calculated over all six MEC nodes for each hour during the entire day. In the figure, the red line represents the MAE for predicting VNF scaling decisions using Q-classifier/C-classifier (1hr) models, and the brown line represents the MAE for predicting scaling decisions using Q-regressor/C-regressor (1hr) models. The former performs better than the latter with respect to MAE throughout the day.

It is worth mentioning that the predicted UPF auto-scaling decisions in Q-regressor/C-regressor (1hr) MLP model (i.e. the best performing model) is used as input to evaluate the SFC placement model presented in Section 5. However, in doing so, we assume that UEs can be associated and served by any of the cells of a candidate gNodeB, to simplify the problem.

5. Latency-optimal SFC placement problem description and network model

In this section, we first define the latency-optimal SFC placement problem and then describe the 5G mobile network model, SFC request model, and UE association, scheduling and delay model employed in formulating the ILP problem.

5.1. Problem statement

Consider a 5G mobile network, composed of *six* gNodeBs, *two* aggregation points, and *one* 5GC, as depicted in Fig. 9. A set of three gNodeBs are interconnected to each other through *Xn*-interfaces. Using *NG*-interfaces, the six gNodeBs are served by two aggregation points, and the 5GC serves both of these aggregation

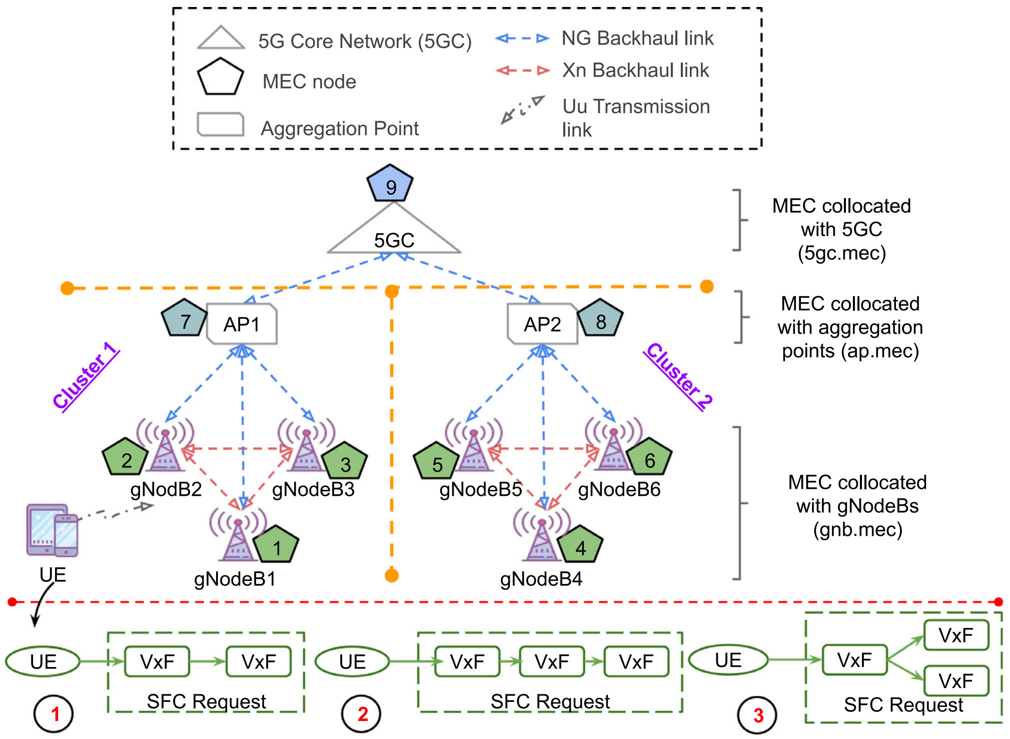

Fig. 9. Substrate network topology and SFC requests.

points. For simplicity, in the rest of this paper, we consider gNodeB1, gNodeB2, gNodeB3, and AP1 belong to cluster 1 while gNodeB4, gNodeB5, gNodeB6, and AP2 belong to cluster 2, as represented in Fig 9. Each element in our network topology is equipped with a resource-constrained (e.g., CPU) MEC node that is capable of hosting SFCs composed of one or several VxFs (e.g., VMs, containers). We consider three feasible options for physically deploying MEC nodes in 5G networks, as defined by ETSI [5], i.e., MEC collocated with gNodeB, MEC collocated with aggregation point, and MEC collocated with 5GC. Furthermore, in the considered hierarchical network topology, we assume that the closer is the MEC node to UE, the less is its computational capacity (e.g., MEC1, MEC2, and MEC3 are identical nodes with least capacity, MEC7 has the medium capacity, and MEC9 has the highest capacity).

Suppose the UE (e.g., autonomous car) is associated with gNodeB2 and requests for an SFC with an end-to-end latency (i.e., real-time, near real-time or non-real-time) and data rate requirements. The SFC requests considered in our work can be either of the three types as depicted in Fig. 9. Depending on the selected cost function to be minimized, the network provider can choose to place the VxFs of the SFC requested by the UE on either the host node (i.e., MEC2) or any neighboring nodes (i.e., MEC1, MEC3) or distant nodes (MEC7, MEC9) or cluster 2 nodes (i.e., MEC4, MEC5, MEC6, MEC8) by allocating sufficient network resources (e.g., CPU, backhaul bandwidth), efficiently, while also making sure that the end-to-end latency and data rate requirements of the requested SFC is always satisfied. In the first case, no additional delay is introduced in the backhaul since the MEC node collocated with the host gNodeB is the one hosting the VxFs. Conversely, in the other three cases, backhaul delay is introduced to map the virtual link onto a backhaul path, connecting the host gNodeB with a neighboring MEC node or a distant MEC node or a MEC node from a different cluster that is hosting VxFs. Formally, the problem of latency-optimal SFC placement is stated as follows:

Given: a small 5G mobile network with gNodeBs, aggregation points, 5GC, MEC nodes, the scheduling capabilities of gNodeBs (e.g., PRBs, TTI duration, subcarrier spacing), the computational capacity of each MEC node, the transport network topology with the capacity of each backhaul link, the number of UEs and their requested SFCs with an end-to-end latency and data rate requirements.

Find: *'where'* to allocate resources to VxFs and *'which'* network paths to use.

Objective: minimize average end-to-end latency for UEs to access their SFCs in the mobile network.

5.2. 5G mobile network model

The mobile network infrastructure is modeled as an undirected graph $G_{net} = (N_{net}, E_{net})$, where $N_{net} = N_{gnb.mec} \cup N_{ap.mec} \cup N_{5gc.mec}$ is the union of the set of $|N_{gnb.mec}|$ gNodeBs collocated with the MEC node, $|N_{ap.mec}|$ aggregation points collocated with the MEC node, and $|N_{5gc.mec}|$ 5GCs collocated with the MEC node and E_{net} is the set of backhaul links such that an edge $e^{mn} \in E_{net}$ only if a connection exists between $m, n \in N_{net}$. Each network node $m \in N_{net}$ is attributed with a weight $w^{net}_{cpu}(m)$, representing its CPU capacity, under the assumption that one VxF requires one CPU unit to be instantiated. Additionally, each network node $m \in N_{gnb.mec}$ is also associated with a weight $w^{gnb.mec}_{prb}(m)$ representing the number of Physical Resource Blocks (PRBs) available at each timeslot that can be scheduled to UEs for trasmitting data packets. Furthermore, each edge $e^{mn} \in E_{net}$ is associated with a weight $w^{net}_{bw}(e^{mn})$ representing its bandwidth capacity (in Gbps). Finally, each network node $m \in N_{net}$ is associated with a geographical location $loc(m)$ (in terms of (x, y) coordinates) and each network node $m \in N_{gnb.mec}$ is associated with a coverage area $cov(m)$. Table 6 summarizes all the parameters used in the mobile network model.

5.3. Service function chain request model

Let $G_{req} = (N_{req}, E_{req})$ be a directed graph modeling the SFC requests, where $N_{req} = N_{ue} \cup N_{sfc}$ is the union of the set of $|N_{ue}|$ UEs

Table 6
Parameters in the mobile network model.

Notation	Definition
G_{net}	Graph of the mobile network.
N_{net}	Set of all network nodes in G_{net}.
$N_{gnb.mec}$	Set of gNodeBs collocated with the MEC node in G_{net}.
$N_{ap.mec}$	Set of aggregation points collocated with the MEC node in G_{net}.
$N_{5gc.mec}$	Set of 5GCs collocated with the MEC node in G_{net}.
E_{net}	Set of all backhaul links in G_{net}.
$w_{cpu}^{net}(m)$	Computing capacity of the network node $m \in N_{net}$.
$w_{prb}^{gnb.mec}(m)$	PRBs available for each timeslot at gNodeB $m \in N_{gnb.mec}$.
$w_{bw}^{net}(e^{mn})$	Bandwidth capacity of the backhaul link $e^{mn} \in E_{net}$.
$loc(m)$	Geographical location of the network node $m \in N_{net}$.
$cov(m)$	Coverage area of the gNodeB $m \in N_{gnb.mec}$.

Table 7
Parameters in the SFC request model.

Notation	Definition
G_{req}	Graph of the SFC association request.
N_{req}	Set of all UEs and their SFC requests in G_{req}.
N_{ue}	Set of UEs in $Greq$.
$Nsfc$	Set of all the SFCs in G_{req}.
$Nvnfs$	Set of all the VxFs available to compose an SFC.
E_{req}	Set of all virtual links in G_{req}.
$D_{E2E,max}(u, s)$	Maximum acceptable end-to-end latency for a UE $u \in N_{ue}$ on its requested service $s \in N_{sfc}$.
$Thr_{req}(u, s)$	Requested data rate for a UE $u \in N_{ue}$ on the requested service $s \in N_{sfc}$.
$loc(u)$	Geographical location of the UE $u \in N_{ue}$.

and $|N_{sfc}|$ the set of SFCs requested from the UEs and E_{req} is the set of virtual links between the UEs and their requested SFCs. Each SFC $s \in N_{sfc}$ is composed of a UPF (i.e., for encapsulation and decapsulation of GPRS Tunnelling Protocol for the user plane (GTP-U) of an UE requesting MEC services [5]) and one or more VMAFs from a set of N_{vnfs}. Each SFC $s \in N_{sfc}$ is characterized by a maximum acceptable end-to-end latency (e.g., real-time, near real-time, non real-time) represented by $D_{E2E,max}(u, s)$ and a minimum guaranteed data rate denoted by $Thr_{req}(u, s)$ that needs to be satisfied. Each UE $u \in N_{ue}$ is associated with a location $loc(u)$ (in terms of (x, y) coordinates). Table 7 summarizes the parameters used in the SFC request model.

5.4. UE association, scheduling and delay model

In contrast to 4G technology where the goal is only to enhance the throughput of Mobile Broadband (MBB) services, 5G is expected to support low-latency applications with end-to-end latency constraints of $1 - 10ms$ and error rates of 10^{-3} to 10^{-5} (e.g., connected cars). For cellular communicatons, two types of latencies are defined in 3GPP: control-plane (C-plane) latency and user-plane (U-plane) latency. The C-plane latency is the transition time for the UE to switch from idle mode to connected mode including the establishment of the user plane while U-plane latency is the one-way delay required to transmit a data packet from the UE to the mobile network (uplink) or vice-versa (downlink) [27].

In this work, we consider only the U-plane latency for computing end-to-end delay (D_{E2E}), since it is the major contributor that is hindering the support of URLLC applications. D_{E2E} is computed from the time UEs start transmitting packets in uplink untill the time they start being processed in MEC nodes. For a scheduled UE, we assume we have 3 different communication delays contributing to D_{E2E}:

(i) ***Radio delay*** ($D_{radio}^{ue.gnb}$) is the sum of UE processing delay (t_{ue}^{proc}), over-the-air transmission delay (t_{TTI}), gNodeB processing delay (t_{gnb}^{proc}), scheduler queuing delay (t_q), and HARQ retransmission delay, which is given by Eq. (13),

$$D_{radio}^{ue,air,gnb} = t_{ue}^{proc} + t_{TTI} + t_{gnb}^{proc} + t_q \\ +2.n_{harq}(t_{ue}^{proc} + t_{TTI} + t_{gnb}^{proc} + t_q) \quad (13)$$

where n_{harq} is the number of HARQ retransmissions required to achieve a BLER target of 10^{-3} to 10^{-5}. Similar to 3GPP, we adopt Orthogonal Frequency Division Multiplexing (OFDM) scheme. To satisfy the latency requirements of URLLC, 3GPP also proposes new frame structures with shorter TTI durations and multiple subcarrier spacings. Scaling up the base subcarrier spacing of $15kHz$ by 2^{μ} (e.g., $30kHz$, $60kHz$, and $120kHz$), the TTI duration of $1ms$ is scaled down by 2^{μ} (e.g., $0.5ms$, $0.25ms$, and $0.125ms$), where $\mu = \{1, 2, .., n\}$, enabling faster transmission and lower processing time [28]. In our model, we adopt a TTI duration of $0.25ms$ resulting in a subcarrier spacing of $60kHz$ for all URLLC UEs. The resulting t_{ue}^{proc} and t_{gnb}^{proc} processing delays are 3 OFDM symbols and 1 TTI, respectively, as measured in [29]. The scheduler queuing delay (t_q), as represented in Eq. (14), is the sum of offset time (t_{offset}) i.e., the waiting time (~ 0 to 1 TTI) once the packet is ready for transmission until the beginning of the next TTI and the packet congestion time (t_{pktcon}) i.e., if the scheduler does not have enough PRBs to schedule a requested SFC packet in one TTI, the SFC packets may remain in the gNodeB buffer for longer duration.

$$t_q = t_{offset} + t_{pktcon} \quad (14)$$

To determine t_{pktcon}, we first need to determine the number of PRBs ($N_{prb}(u, s, m)$) required for the SFC $s \in N_{sfc}$ of an UE $u \in N_{ue}$ to be assigned by its associated gNodeB $m \in N_{gnb.mec}$. Given the data rate demand of the SFC s, the number of PRBs ($N_{prb}(u, s, m)$) is computed according to Eq. (15) as given in 3GPP [30]:

$$N_{prb}(u, s, m) = \frac{Thr_{req}(u, s) * T_s^{\mu}}{12 * 10^{-6} * N_{cc} * N_{mimo} * N_{mod} * sf * R * (1 - oh)} \quad (15)$$

where, N_{cc} is the number of aggregated component carriers, N_{mimo} is the number of MIMO layers, N_{mod} is the modulation order (e.g., 2 for QPSK, 4 for 16QAM, 6 for 64QAM, 8 for 256QAM), sf is the scaling factor, R is the code rate, oh is the overhead for control channels, and $T_s^{\mu} = 10^{-3}/(14 * 2^{\mu})$ is the average OFDM symbol duration in a subframe for numerology μ ($\mu = 2$ in our case) assuming normal cyclic prefix. Except for N_{mod} and R, which are determined as per the below three steps, all other parameters are predefined according to the radio access capabilities:

SINR measurement: The UE $u \in N_{ue}$ measures the SINR value for a reference signal coming from its associated gNodeB $m \in N_{gnb.mec}$ using Eq. (16),

$$sinr(u, m) = \frac{\frac{F_m}{|loc(u) - loc(m)|^{\alpha}}}{\sum_{m' \neq m} \frac{F_{m'}}{|loc(u) - loc(m')|^{\alpha}} + N} \quad (16)$$

where $|loc(u) - loc(m)|$ is the distance between the UE u and its associated gNodeB m, $|loc(u) - loc(m')|$ is the distance between the UE u and the neighbouring gNodeBs of m (m'), α is a path loss exponent between 2 and 6, F_m and $F_{m'}$ are fading random variables of some distribution, and N is a constant noise term [31].

CQI report: The UE $u \in N_{ue}$ maps the SINR value measured in step 1 to a CQI index from the mapping table in [32], which is expected to be reported to the scheduler of its associated gNodeB $m \in N_{gnb.mec}$. It is to be noted that these mappings are not defined in 3GPP but are vendor specific.

CQI to MCS mapping: The scheduler is now expected to map the reported CQI index to an MCS index and determine the best combination of modulation order (N_{mod}) and code rate (R) to be used from the mapping table in [33], resulting in a BLER target of 10^{-5}.

Therefore, we have all the necessary parameters in Eq. (15) to determine the number of PRBs that must be assigned for an SFC

$s \in N_{sfc}$ requested by the UE $u \in N_{ue}$ to meet its data rate requirements. If the number of PRBs that needs to be assigned are not available in a particular TTI, they will be assigned during the next TTI and so forth, adding up to the total t_q latency (i.e., t_{pktcon} = no. of TTIs to schedule SFC packets * TTI duration).

(ii) ***Backhaul delay*** ($D_{bh}^{Xn,NG}$) is the sum of X_n propagation delay (t_{Xn}^{prop}), X_n transmission delay (t_{Xn}^{tx}), NG propagation delay (t_{NG}^{prop}), and NG transmission delay (t_{NG}^{tx}), given by Eq. (17),

$$D_{bh}^{Xn,NG} = t_{Xn}^{prop} + t_{Xn}^{tx} + t_{NG}^{prop} + t_{NG}^{tx} \tag{17}$$

where for a link $e^{mn} \in E_{net}$, t_{Xn}^{prop} refers to the propagation time required to transmit SFC packets from node $m \in N_{gnb.mec}$ to node $n \in N_{gnb.mec}$ while t_{NG}^{prop} refers to the propagation time required to transmit SFC packets from node $m \in N_{gnb.mec}$ to node $n \in N_{ap.mec}|N_{5gc.mec}$. Similarly, t_{Xn}^{tx} and t_{NG}^{tx} refers to the transmission time required to transfer SFC packets from node $m \in N_{gnb.mec}$ and node $m \in N_{ap.mec}|N_{5gc.mec}$, to the outgoing link e^{mn}, respectively.

(iii) ***SFC processing delay*** (D_{mec}^{sfc}) is the time required for all VxFs in an SFC $s \in N_{sfc}$ to apply a specific network operation on the arriving packets.

Therefore, D_{E2E} is computed according to Eq. (18).

$$D_{E2E} = D_{radio}^{ue,air,gnb} + D_{bh}^{Xn,NG} + D_{mec}^{sfc} \tag{18}$$

It is to be noted that the same delay model can be used for both downlink and uplink direction.

6. Problem formulation

Once, a batch of UE associations and its SFC requests arrive at the substrate network, it is either approved and embedded onto the network or it is denied. The embedding process includes both node and link mapping and is generally referred to as virtual network embedding problem which is proven to be NP-hard [34]. In the node mapping stage, each virtual node (i.e., UEs, VxFs in the SFCs requested by UEs) is mapped to a substrate node (i.e., gNodeBs, MEC nodes) while in the link mapping stage, each virtual link (i.e., the link between the UE and its requested SFC) is mapped to a single substrate path (i.e. the path between the gNodeB hosting the UE and MEC nodes hosting the VxFs in the SFC). In both stages, the constraints imposed on substrate nodes and substrate links must be satisfied.

6.1. Integer linear programming

The proposed joint UE association and SFC placement problem is formulated employing ILP techniques. Before starting the actual problem formulation, for each UE $u \in N_{ue}$, we first determine the set of candidate gNodeBs ($\overline{N}_{gnb.mec}(u)$) using Eq. (19),

$$\overline{N}_{gnb.mec}(u) = \{m \in N_{gnb.mec} | (|loc(u) - loc(m)|) \leq cov(m)\} \tag{19}$$

Then, we find neighboring gNodeBs for each gNodeB $m \in N_{gnb.mec}$ and neighboring MEC nodes for each MEC node $m \in N_{net}$ using Eqs. (20) and (21), respectively.

$$nbr\_gnbs(m) = \{m' \in N_{gnb.mec} | e^{m,m'} \in E_{net}\} \tag{20}$$

$$nbr\_nodes(m) = \{m' \in N_{gnb.mec}, m'' \in N_{ap.mec}, \times m''' \in N_{5gc.mec} | e^{m,m'}, e^{m,m''}, e^{m'',m'''} \in E_{net}\} \tag{21}$$

Next, we find the candidate MEC nodes that can host VxFs of SFC requested by UEs. For each VxF v of SFC s from the UE u, the set of candidate MEC Nodes ($\overline{N}_{net}(u, v, s)$) can be defined according to Eq. (22).

$$\overline{N}_{net}(u, v, s) = \{m \in \overline{N}_{gnb.mec}(u), m' \in nbr\_gnbs(m), m'' \in N_{ap.mec}, \times m''' \in N_{5gc.mec} | e^{m,m'}, e^{m,m''}, e^{m'',m'''} \in E_{net}\} \tag{22}$$

Thus, in our ILP model, either the UEs candidate gNodeB MEC node, or the gNodeB MEC node connected to the candidate gNodeB MEC node, or the aggregation point MEC node connected to the candidate gNodeB MEC node, or the 5GC MEC node connected to the aggregation point MEC node serving the candidate gNodeB MEC node can host UEs SFC.

Now, we formulate the SFC placement problem with three binary decision variables, χ_m^u, $\Upsilon_m^{u,v,s}$, and $\Psi_{m,n}^{u,s}$, as represented in Table 8.

Table 8
Binary decision varibles.

Notation	Definition
χ_m^u	To show if $u \in N_{ue}$ is associated to gNodeB $m \in N_{gnb.mec}$.
$\Upsilon_m^{u,v,s}$	To show if $v \in N_{vnfs}$ of $s \in N_{sfc}$ from $u \in N_{ue}$ is assigned to $m \in N_{net}$.
$\Psi_{m,n}^{u,s}$	To show if virtual link between $u \in N_{ue}$ and $s \in N_{sfc}$ is assigned to substrate link between $m \in N_{net}$ and $n \in nbr\_nodes(m)$.

The objective function of the ILP, given in Eq. (23), is to minimize the overall end-to-end latency from all users to their respective SFCs.

$$\begin{aligned} ILP: min[& \sum_{u \in N_{ue}} \sum_{m \in N_{gnb.mec}} \chi_m^u * D_{radio}^{ue,air,gnb}(u, m) \\ & + \sum_{u \in N_{ue}} \sum_{v \in N_{vnfs}} \sum_{s \in N_{sfc}} \sum_{m \in N_{net}} \Upsilon_m^{u,v,s} * D_{mec}^{sfc}(u, v, s, m) \\ & + \sum_{u \in N_{ue}} \sum_{s \in N_{sfc}} \sum_{m \in N_{net}} \sum_{n \in nbr\_nodes(m)} \Psi_{m,n}^{u,s} * D_{bh}^{Xn,NG}(u, s, m, n)] \end{aligned} \tag{23}$$

In Eq. (23), $D_{radio}^{ue,air,gnb}(u, m)$ depends on the number of UEs that need to be scheduled in a given time slot by gNodeB m. For each UE u, we first find the $sinr(u, m)$ from Eq. (16) and then calculate the needed $N_{prb}(u, s, m)$ from Eq. (15) to transmit packets of SFC s with a particular size at a requested data rate ($Thr_{req}(s)$). If the total required PRBs exceed the maximum available PRBs in the gNodeB, some UEs are scheduled in the next time slot, thus increasing the Radio delay for those UEs. Since the backhaul links, Xn and NG, in the mobile network, $D_{bh}^{Xn,NG}(u, s, m, n)$ depends on the number of UEs sharing the same backhaul link.

We will now describe all node and link constraints imposed in our problem formulation. Constraint (24) ensures that each UE is associated to only one gNodeB from its candidate set.

$$\sum_{m \in \overline{N}_{gnb.mec}(u)} \chi_m^u = 1, \forall u \in N_{ue} \tag{24}$$

Constraint (25) guarantees that each VxF of SFC requested from each UE is hosted by only one substrate MEC node from its candidate set.

$$\sum_{m \in \overline{N}_{net}(u,s)} \Upsilon_m^{u,v,s} = 1, \forall u \in N_{ue} \forall s \in N_{sfc}^u \forall v \in N_{vnfs}^s \tag{25}$$

Constraint (26) guarantees that each VxF is at most shared by vnf_{max}^{shared} number of UEs.

$$\sum_{v \in N_{vnfs}} \Upsilon_m^{u,v,s} \leq vnf_{max}^{shared}, \forall u \in N_{ue} \forall s \in N_{sfc}^u \quad \forall m \in \overline{N}_{net}(u, s) \tag{26}$$

Constraint (27) ensures that the amount of CPU resources allocated to VxFs of SFCs adheres to the available CPU capabilities on the substrate node.

$$\sum_{u \in N_{ue}} \sum_{s \in N^u_{sfc}} \sum_{v \in N^s_{vnfs}} \Upsilon^{u,v,s}_m \le w^{net}_{cpu}(m), \forall m \in N_{net} \tag{27}$$

Constraint (28) makes sure that in each time slot gNodeBs can associate UEs only if they have enough PRBs to meet the data rate demand of the requested SFC by the UE.

$$\sum_{u \in N_{ue}} \sum_{s \in N_{sfc}} N_{prb}(u,s,m) * \chi^u_m \le w^{gnb.mec}_{prb}(m), \quad \forall m \in N_{gnb.mec} \tag{28}$$

Flow constraint (29) enforces for each virtual link between UE $u \in N_{ue}$ and its SFC $s \in N_{sfc}$ there exists a continuous path established between the gNodeB to which the UE is associated and the MEC node hosting the VxFs of SFC s.

$$\sum_{n \in nbr\_nodes(m)} (\Psi^{u,s}_{n,m} - \Psi^{u,s}_{m,n}) = \Upsilon^{u,s}_m - \chi^u_m, \quad \forall m \in N_{net}, \forall e^{u,s} \in E_{req} \tag{29}$$

Constraint (30) makes sure that virtual links are mapped onto the backhaul substrate links in the mobile network, if and only if it has enough bandwidth capacity to meet the link demand of virtual links.

$$\sum_{u \in N_{ue}} \sum_{s \in N_{sfc}} Thr_{req}(u,s)(\Psi^{u,s}_{n,m} + \Psi^{u,s}_{m,n}) \le w^{net}_{bw}(e^{nm}),$$
$$\forall m \in N_{net}, \forall n \in nbr\_nodes(m), n < m \tag{30}$$

Constraint (31) ensures that the end-to-end latency from the UEs to its associated SFCs does not exceed the maximum acceptable latency as requested by the UEs.

$$\sum_{m \in N_{gnb.mec}} \chi^u_m * D^{ue,air,gnb}_{radio}(u,m)$$
$$+ \sum_{m \in N_{net}} \sum_{v \in N^s_{vnfs}} \Upsilon^{u,v,s}_m * D^{sfc}_{mec}(u,v,s,m)$$
$$+ \sum_{m \in N_{net}} \sum_{n \in nbr\_nodes(m)} \Psi^{u,s}_{m,n} * D^{Xn,NG}_{bh}(u,s,m,n) \le D_{E2E,max}(u,s),$$
$$\forall u \in N_{ue}, \forall s \in N^u_{sfc} \tag{31}$$

6.2. Heuristic

The above ILP formulation took 44 hours to associate 300 UEs including their latency-sensitive SFC requests composed of a number of VxFs on a mobile network comprised of *six* gNodeBs, *two* aggregation points, and *one* 5G core. The ILP was solved using ILOG CPLEX solver on an Intel Core i7 laptop with 3GHz CPU and 16 GB RAM. To address the issue of scalability in ILP, we propose a heuristic algorithm, as seen in Algorithm 1, that can solve the above association/mapping problem in a couple of seconds. Similar to the ILP-based algorithm, the objective of our heuristic algorithm is to minimize the overall end-to-end latency from all UEs to their requested SFCs.

In the first step (lines 1 – 10), the algorithm loops through all the UEs to determine a set of candidate gNodeBs considering the location of the UE, location of the gNodeB, and the coverage area of the gNodeB, and then creates a list of $cand\_gnb(u)$ for each UE. Next, each UE is mapped to the gNodeB, among the $cand\_gnb(u)$, that measures the best signal quality (i.e., SINR) and also has sufficient PRBs to host the UE.

In the second step (lines 11 – 20), the algorithm finds the candidate MEC nodes for each VxF of SFC requests received from all

Algorithm 1 Heuristic.

```
Require: $G_{net}$, $G_{req}$, and SFC latency budget [$N_{sfc}(rt)$, $N_{sfc}(near\_rt)$, $N_{sfc}(non\_rt)$].
Ensure: User association and latency-optimal SFC placement.
    Step 1. Find candidate gNodeBs for each UE and perform UE association.
 1: for $u$ in $N_{ue}$ do
 2:     $cand\_gnb(u) \leftarrow 0$
 3:     $map\_gnb(u) \leftarrow 0$
 4:     for $m$ in $N_{gnb.mec}$ do
 5:         if $|loc(u) - loc(m)| <= cov(m)$ then
 6:             $cand\_gnb(u) \leftarrow m$
 7:         end if
 8:     end for
 9:     $map\_gnb(u) \leftarrow m$ from the list of $cand\_gnb(u)$ with $max(sinr(u,m))$ and enough PRBs available.
10: end for
    Step 2. Find candidate MEC nodes for VxFs of each SFC from each UE.
11: for $u$ in $N_{ue}$ do
12:     for $s$ in $N_{sfc}$ do
13:         for $v$ in $N_{sfc}(u)$ do
14:             $cand\_mec(u,s,v) \leftarrow 0$
15:             for $m$ in $neighbours(map\_gnb(u))$ do
16:                 $cand\_mec(u,s,v) \leftarrow m$
17:             end for
18:         end for
19:     end for
20: end for
    Step 3. Perform SFC placement for each UE.
21: for $u$ in $N_{ue}$ do
22:     for $s$ in $N_{sfc}(rt)$ do /* real-time SFCs. */
23:         for $v$ in $N_{sfc}(u)$ do
24:             $map\_mec(u,s,v) \leftarrow 0$
25:             for $m$ in $cand\_mec(u,s,v)$ do
26:                 $compute(D_{E2E}(u,s,m))$
27:                 if $D_{E2E}(u,s,m) <= D_{E2E,max}$ then
28:                     if $inst(v)$ not in $m$ or $neighbours(m)$ then
29:                         $map\_mec(u,s,v) \leftarrow m$
30:                     end if
31:                 end if
32:                 $alocate\_continuous\_path(u,s,m)$
33:                 $update\_node\_and\_link\_resources()$
34:             end for
35:         end for
36:     end for
37: end for
38: Repeat Step 3 for s in $N_{sfc}(near\_rt)$ and $N_{sfc}(non\_rt)$.
```

UEs, which is nothing but the union of the MEC node collocated with the host gNodeB of UE (determined from step 1) and all other MEC nodes connected directly or indirectly to the host gNodeB of UE through backhaul links. Another list of $cand\_mec(u,s,v)$ is created for each VxF of the SFC received from all UEs.

In the third step (lines 21 – 38), the algorithm begins mapping all VxFs of SFC requests with real-time latency requirements considering $cand\_mec(u,s)$ for each SFC, starting from *gnb.mec* nodes. Once they run out of computing resources, the algorithm moves on to *ap.mec* nodes, and finally on to 5*gc.mec* nodes, if and only if the computed end-to-end latency ($D_{E2E}(u, s, m)$) is less than the maximum acceptable end-to-end latency for that SFC ($D_{E2E,max}$). Moreover, if an instance of VxF is already mapped to the candidate MEC Node the UE shares this VxF to realize its SFC instead of instantiating a new VxF. The heuristic then uses the shortest path algorithm to map the virtual link between the UE and its requested SFC onto the substrate link between the gNodeB that the UE is associated to and the MEC node that the SFC is being hosted on. The VxFs of a single SFC might be mapped on different MEC nodes, and therefore further caution is exercised during link mapping. The node and link computational resources are updated after each mapping. The same process is repeated for other SFC requests with near-real-time and non-real-time latency requirements until all SFC requests are mapped.

7. ILP and Heuristic evaluation

The performance of the latency-optimal SFC placement ILP model is evaluated based on the simulations implemented in Python. We then compare it to the implemented heuristic algorithms performance. Real-operator network topology and realistic latency values are used when modeling the simulation environment to produce realistic simulation results, which can better illustrate the benefits of placing SFCs composed of VxFs at the network edges closer to the end-user.

7.1. Simulation environment

A small cluster of 5G mobile network composed of 9 network nodes is considered in our simulation, as depicted in Fig. 9. A set of 3 gNodeBs are connected to each other through 20 Gbps *Xn* backhaul links, while each of the three gNodeBs is connected to the aggregation point using 20 Gbps *NG* backhaul links which in turn is connected to the 5G core network using 50 Gbps backhaul links. The number of aggregated component carriers is set to 4, and each carrier has a bandwidth capacity of 20 MHz. We assume that the gNodeBs support 4*x*4 MIMO configuration. We then introduce MEC nodes at each of these 9 network nodes capable of hosting a limited number of VxFs. The MEC nodes collocated with gNodeBs each have 50 CPUs, the MEC nodes collocated with aggregation points each have 100 CPUs, and the MEC node collocated with 5GC has 500 CPUs.

Our simulations are carried out for two scenarios. In the first scenario, we consider that SFC requests arrive in batches of 30 UEs (equally divided among real-time, near-real-time, and non-real-time) with each batch corresponding to 1 timeslot. In every timeslot, the ILP considers the SFC requests received in previous batches and associates all the UEs and their SFC requests onto the mobile network, considering the latency and data rate requirements of each SFC. We consider 10 batches of SFC requests corresponding to 300 UEs. In the second scenario, we consider that SFC requests arrive according to the *predicted number of UPF instances from our MLP neural-network model*, as illustrated in Section 4. The performance of ILP is compared with our heuristic algorithm in both scenarios. Additionally, in both scenarios, we assume that each UPF instance corresponds to one UE and each SFC is composed of 2 or 3 VxFs (1 UPF and 1 or 2 VMAF as depicted in Fig. 9) with 1 CPU required to instantiate every VxF. Furthermore, each UPF is shared with 5 UEs while each VMAF is shared among 2 UEs. Since the UEs considered in our model are URLLC UEs, we assume three categories of user-to-SFC one-way delay requirements, i.e., 1*ms*, 2*ms*, and 5*ms*. We assume that each UE is transmitting short packets of size 15*Kb* every *TTI* and requests a minimum data rate of 200*Mbps*. In Section 5, we discussed on how we calculate $D^{ue,air,gnb}_{Radio}$. We calculate $D^{Xn,NG}_{bh}$ by dividing the total packet size generated from all the UEs that are using the same backhaul link with the bandwidth capacity of the link [35]. Finally, we calculate D^{sfc}_{mec} by dividing the packet size that the VxFs of the SFC should process by the CPU speed. We consider a CPU speed of 3*GHz* with 64 bit processor.

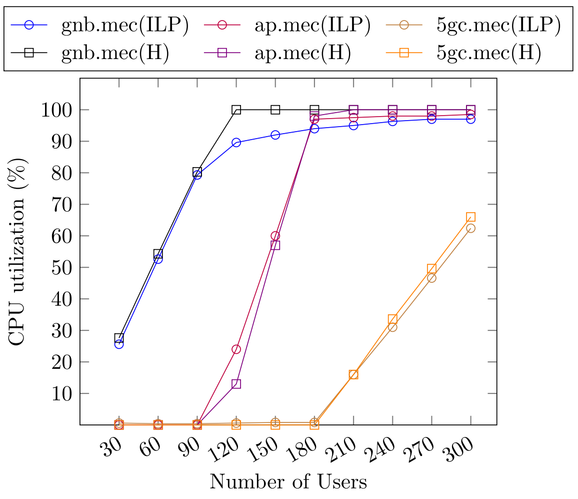

Fig. 10. CPU utilization of MEC nodes (Scenario 1).

7.2. Simulation results

CPU Utilization: The CPU utilization is computed by dividing the number of CPUs utilized in *gnb.mec* nodes or *ap.mec* nodes or 5*gc.mec* nodes once the VxFs are mapped to the total number of CPUs available in all *gnb.mec* nodes or all *ap.mec* nodes or all 5*gc.mec* nodes, respectively.

Fig. 10 illustrates the CPU utilization of MEC nodes with respect to the number of UEs for simulations carried out in scenario 1. We observe that up to ≈90 UEs, the ILP places most of the VxFs on *gnb.mec* nodes because of its proximity to UEs, irrespective of the SFC latency requirements, while some non-real-time VxFs that are shared by UEs associated with cluster 1 (gNodeB1, gNodeB2 or gNodeB3) and cluster 2 (gNodeB4, gNodeB5 or gNodeB6) are placed on 5*gc.mec* nodes. Only after *gnb.mec* nodes are depleted with their CPU resources (after 90 UEs), the ILP starts moving VxFs with near-real-time and non-real-time latency requirements initially placed in *gnb.mec* nodes to *ap.mec* nodes and starts placing new SFCs with real-time latency requirements on *gnb.mec* nodes. Similarly, when CPU resources of *ap.mec* nodes are depleted (≈180 UEs) the ILP starts moving VxFs with non-real-time latency requirements initially placed in *gnb.mec* or *ap.mec* nodes to 5*gc.mec* nodes. On the other hand, heuristic algorithm follows a similar pattern to that of ILP, but instead of placing non-real-time VxFs that are shared by UEs associated to cluster 1 and cluster 2 on 5*gc.mec* nodes, those VxFs are initially placed on *gnb.mec* nodes, then on *ap.mec* nodes and finally on 5*gc.mec* nodes(in this order depending on the MEC nodes resource availability). This is evident from Fig. 10, where CPU utilization of *gnb.mec* nodes is always higher in heuristic compared to that of ILP. However, this results in the increase of overall latency for heuristic due to the users taking a long path to access their SFC services (e.g., if a user is associated to gNodeB2 and its VxFs are placed in *gNB*6.*mec*, the path mapping could be gNodeB2 → AP1 → 5GC → AP2 → gNodeB6). Consequently, up to 180 UEs, the CPU utilization of *ap.mec* and 5*gc.mec* nodes are lower in heuristic compared to ILP.

Fig. 11 illustrates the CPU utilization of MEC nodes with respect to time over one full day for the network considered in the MLP neural-network model (scenario 2). We observe that the CPU utilization of *gnb.mec* nodes are always full, the *ap.mec* nodes are most of the time full except from 2:00 to 8:00 and 5*gc.mec* nodes have low utilization during early morning (2:00 to 8:00) due to the low number of UEs being active and the utilization gradually increases during the day peaking late in the night (≈ 22:00). The heuristic follows a similar pattern to that of ILP, but as discussed

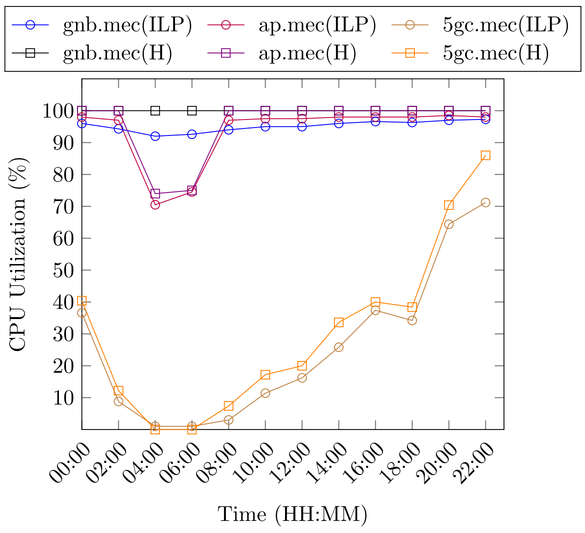

Fig. 11. CPU utilization of MEC nodes (Scenario 2).

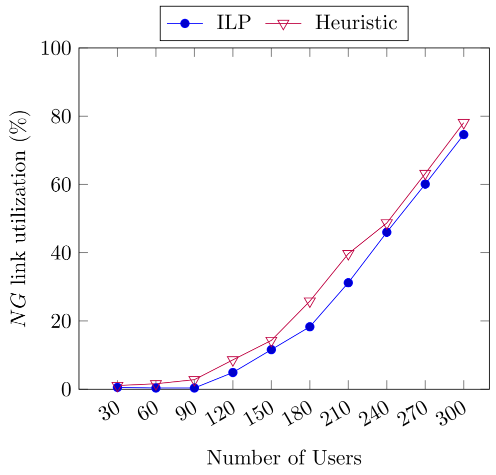

Fig. 13. NG-link (gNodeB-to-AP-to-5GC) utilization (Scenario 1).

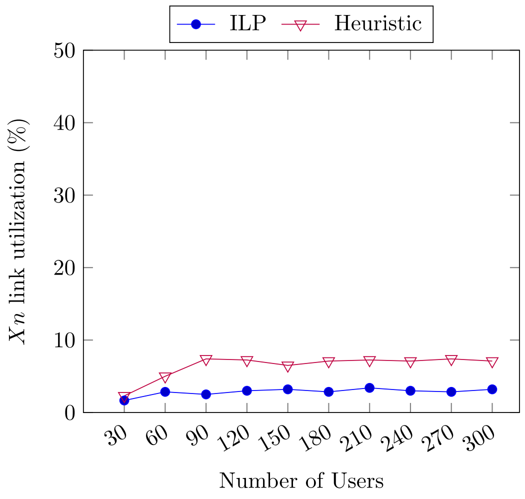

Fig. 12. Xn-link (gNodeB-to-gNodeB) utilization (Scenario 1).

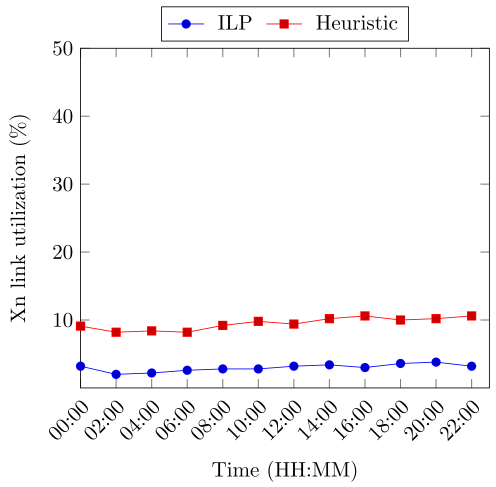

Fig. 14. Xn-link (gNodeB-to-gNodeB) utilization (Scenario 2).

earlier VxFs shared by UEs belonging to different clusters are initially placed on *gnb.mec* nodes rather than on *5gc.mec* nodes like in ILP. Therefore, CPU utilization for *gnb.mec* nodes are always higher in heuristic compared to that of ILP, with a tradeoff being the increase in overall latency.

Link utilization: Link utilization is calculated by dividing the usage of either *Xn* or *NG* backhaul links by UEs for utilizing SFCs in the MEC nodes to the total available capacity of the respective links.

Figs. 12 and 13 illustrates, respectively, the *Xn* link utilization and the *NG* link utilization as a function of the number of UEs for experiments carried out in scenario 1. In Fig. 12, we observe that in ILP, irrespective of the number of UEs, the *Xn* link utilization remains almost the same ($<3\%$), which is attributed to the fact that ILP principally places the VxFs of UEs on the *gnb.mec* that is currently serving the corresponding UE over the air interface in order to minimize the end-to-end latency. However, we observe that heuristic algorithm places VxFs of some UEs on *gnb.mec* nodes that are currently not serving the corresponding UE over the air interface, which leads to the usage of *Xn* links. After a certain point ($\approx$90 UEs in Fig. 12), the *Xn* link utilization remains almost the same for heuristic since the capacity of all *gnb.mec* nodes are depleted, and VxFs are placed on *ap.mec* or *5gc.mec* nodes there on.

In Fig. 13, we can observe that both in ILP and heuristic *NG* links are least utilized up to $\approx$90 UEs since most VxFs of SFCs, irrespective of their latency demands, are always placed on *gnode.mec* nodes until then. Once *gnode.mec* nodes are out of CPU resources, the VxFs of SFCs are moved to *ap.mec* nodes and later to *5gc.mec* nodes considering the latency requirements of SFCs, resulting in the significant usage of *NG* backhaul links. However, the reason for higher *NG* link utilization in heuristic is attributed to the fact that some UEs take longer routes, from cluster 1 to cluster 2 or viceversa, in order to access their SFC which is not the case in ILP.

Fig. 14 and Fig. 15 illustrates, respectively, the *Xn* link utilization and the *NG* link utilization with respect to time over one full day based on the network considered in the MLP neural-network model (Scenario 2). Since the number of UEs is always more than 90, we observe that both ILP and heuristic algorithm places VxFs of some UEs on *gnb.mec* nodes that are currently not serving the corresponding UE over the air interface which leads to the usage of *Xn* links as already explained in Scenario 1. Likewise, *NG* link utilization for both ILP and heuristic is lowest during early morning (2:00 to 8:00) due to the low number of UEs being active and the utilization gradually increases during the day peaking late in the night ($\approx$ 22:00). However, both *Xn* and *NG* link utilizations in heuristic are higher compared to ILP because of the long path the UEs take to access SFC like we discussed before.

Average end-to-end latency: Fig. 16 compares the average user-to-sfc end-to-end delay for ILP and heuristic for Scenario 2 ex-

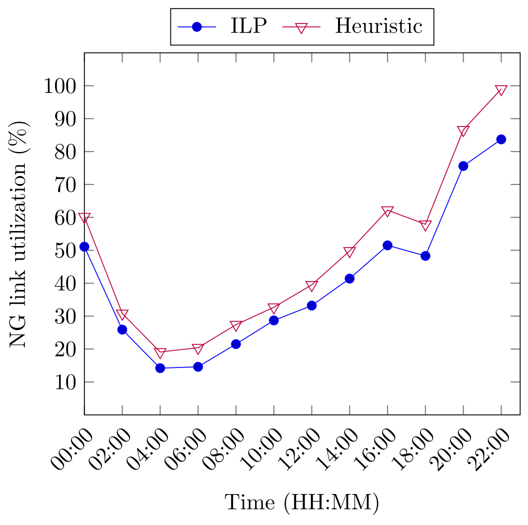

Fig. 15. NG-link (gNodeB-to-AP-to-5GC) utilization (Scenario 2).

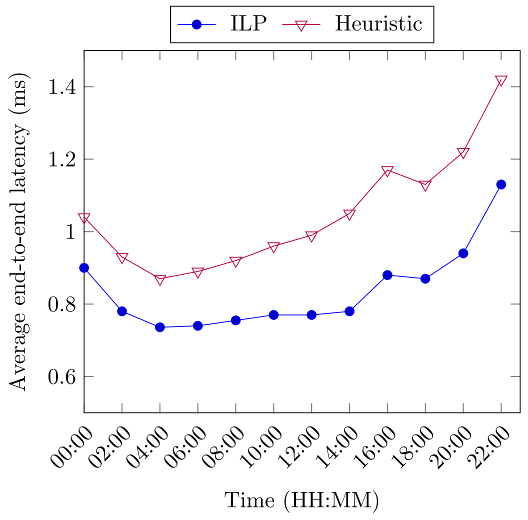

Fig. 16. Average D_{E2E} based on the predicted number of UPFs from the MLP classifier model (Scenario 2).

periments. Like we already discussed, UEs belonging to different clusters share some VxFs. The ILP produces an optimal solution by placing such VxFs at 5*gc.mec* nodes to minimize the overall user-to-sfc delay, but the heuristic initially places such VxFs on *gnb.mec* nodes, and therefore some UEs take the longer path (e.g., from cluster 1 to cluster 2) to access their SFC resulting in increased latency. Therefore, ILP performs better than heuristic in terms of average end-to-end delay between, as seen in Fig 16.

Execution time: The above ILP formulation took 44 hours to associate 300 UEs including their latency-sensitive SFC requests composed of a number of VxFs on a mobile network comprised of six gNodeBs, two aggregation points, and one 5G core. Therefore, we proposed a heuristic algorithm that performs a comparable association and mapping in a couple of seconds except with sub-optimal outcomes. Both ILP and heuristic were solved using CLOG IPLEX solver on an Intel Core i7 laptop with 3GHz CPU and 16 GB RAM.

8. Conclusions

The first part of the paper aims at applying machine learning techniques to optimize network management operations. Towards this end, we proposed two neural-network based MLP models (i.e., a classifier and a regressor) to facilitate proactive auto-scaling of VNFs, based on the traffic traces obtained from a commercial operator. We evaluated the proposed models for its effectiveness in accurately predicting the amount of UPF instances required as a function of the network traffic it should process. For MLP classifier, we measured accuracy, precision, recall, F-measure, and finally reported confusion matrix, while for MLP regressor we measured MSE, MAE, RMSE, and R^2-score. Our results show that both MLP classifier and MLP regressor models have strong predicting capability for auto-scaling. However, MLP regressor outperforms MLP classifier in terms of accuracy.

In the second part of the paper, we solve a joint UE association and SFC placement problem aiming to minimize the overall user-to-sfc end-to-end latency. We have seen that the ILP improves QoS of all UEs by initially placing their SFCs in MEC nodes closer to gNodeBs (*gnb.mec*) and thereby reducing *NG* backhaul link usage. Once the *gnb.mec* node CPU resources are depleted, near-real-time and non-real-time SFCs are moved/placed in MEC nodes closer to aggregation points and 5GC which results in increased usage of *Xn* and *NG* backhaul links. We evaluated the proposed model using simulations based on real-operator network topology and real-world latency values. Our results show that the average end-to-end latency reduces significantly when SFCs are placed at the MEC nodes according to their latency and data rate demands. Furthermore, we propose an heuristic algorithm to address the issue of scalability in ILP, that can solve the above association/mapping problem in seconds rather than hours.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Gupta, R.K. Jha, A survey of 5g network: architecture and emerging technologies, IEEE Access 3 (2015) 1206–1232.
[2] Q. V. Pham, F. Fang, V. N. Ha, M. Le, Z. Ding, L. B. Le, W. J. Hwang, A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art. *arXiv:*1906.08452 *(2019)*.
[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, R. Boutaba, Network function virtualization: state-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 18 (1) (2015) 236–262.
[4] I.F. Akyildiz, S. Nie, S.-C. Lin, M. Chandrasekaran, 5G roadmap: 10 key enabling technologies, Comput. Netw. 106 (2016) 17–48.
[5] MEC in 5G networks, Whitepaper, ETSI, 2018.
[6] Network Functions Virtualization (NFV), Whitepaper, ETSI, 2017.
[7] I. Farris, T. Taleb, H. Flinck, A. Iera, Providing ultra-short latency to user-centric 5g applications at the mobile network edge, Trans. Emerg. Telecommun. Technol. 29 (4) (2018) e3169.
[8] E. Casalicchio, L. Silvestri, Mechanisms for sla provisioning in cloud-based service providers, Comput. Netw, 57 (3) (2013) 795–810.
[9] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, R. Boutaba, Delay-aware vnf placement and chaining based on a flexible resource allocation approach, in: 13th International Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–7.
[10] T. Subramanya, R. Riggio, Machine learning-driven scaling and placement of virtual network functions at the network edges, in: 5th International Conference on Network Softwarization, 2019, 2019.
[11] S. Dutta, T. Taleb, A. Ksentini, Qoe-aware elasticity support in cloud-native 5g systems, in: IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.
[12] G.A. Carella, M. Pauls, L. Grebe, T. Magedanz, An extensible autoscaling engine (ae) for software-based network functions, in: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2016, pp. 219–225.
[13] M.M. Murthy, H. Sanjay, J. Anand, Threshold based auto scaling of virtual machines in cloud environment, in: IFIP International Conference on Network and Parallel Computing, Springer, 2014, pp. 247–256.
[14] C.H.T. Arteaga, F. Rissoi, O.M.C. Rendon, An adaptive scaling mechanism for managing performance variations in network functions virtualization: A case study in an nfv-based epc, in: 2017 13th International Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–7.
[15] T. Lorido-Botran, J. Miguel-Alonso, J.A. Lozano, A review of auto-scaling techniques for elastic applications in cloud environments, Journal of grid computing 12 (4) (2014) 559–592.

[16] A. Bilal, T. Tarik, A. Vajda, B. Miloud, Dynamic cloud resource scheduling in virtualized 5g mobile systems, in: 2016 IEEE Global Communications Conference (GLOBECOM), IEEE, 2016, pp. 1–6.
[17] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, R. Boutaba, Topology-aware prediction of virtual network function resource requirements, IEEE Trans. Netw. Serv. Manage. 14 (1) (2017) 106–120.
[18] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M.J. Hibbett, et al., Knowledge-defined networking, ACM SIGCOMM Comput. Commun. Rev. 47 (3) (2017) 2–10.
[19] S. Agarwal, F. Malandrino, C.-F. Chiasserini, S. De, Joint vnf placement and cpu allocation in 5g, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 1943–1951.
[20] H. Hawilo, M. Jammal, A. Shami, Orchestrating network function virtualization platform: Migration or re-instantiation? in: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), IEEE, 2017, pp. 1–6.
[21] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada–Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (1) (2018) 16.
[22] S. Basterrech, G. Rubino, V. Snášel, Sensitivity analysis of echo state networks for forecasting pseudo-periodic time series, in: 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR), IEEE, 2015, pp. 328–333.
[23] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, Y. Bengio, Theano: A Python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688, 2016.
[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado et al., TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow. org 1(2) (2015).
[25] F. Chollet, Keras, 2015.
[26] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, B. Mukherjee, Auto-scaling vnfs using machine learning to improve qos and reduce cost, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.
[27] *Feasibility study for further advancements for E-UTRA (LTE-Advanced)*, 3GPP TR 36.912 version 10.0 0, Release 10, 2011.
[28] 5G frame structure [whitepaper], Nomor Research, Munich, Germany, 2017.
[29] UP Latency in NR, 3GPP technical contribution R2-1711550, Ericsson, 2017.
[30] 5G; NR; User Equipment (UE) radio access capabilities, 3GPP TS 38.306 version 15.3.0 Release 15, 2018.
[31] F. Massimo, M. Ronald, Random networks for communication: from statistical physics to information systems, 24, Cambridge University Press, 2008.
[32] A. Othman, S.Y. Ameen, H. Al-Rizzo, A new channel quality indicator mapping scheme for high mobility applications in lte systems, J. Model. Simul. Antennas Propag. 1 (2) (2015) 38–43.
[33] Physical layer procedures for data, 3GPP TS 38.214 version 15.3.0 Release 15, 2018.
[34] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, 1998.
[35] D. Harutyunyan, S. Nashid, B. Raouf, R. Riggio, Latency–aware service function chain placement in 5g mobile networks, in: IEEE Conference on Network Softwarization, 2019.

Tejas Subramanya received the bachelor's degree in electronics and communications from BNM Institute of Technology, Bengaluru, India in 2010 and the M.S degree in radio communications from Aalto University, Espoo, Finland in 2014. He is currently pursuing the Ph.D. degree in IT and Telecommunications from the University of Trento, Italy and is also a Researcher with the WiN unit in FBK CREATE-NET, Trento, Italy. Before completing his M.S degree, he was working as a Research and Development Engineer with Nokia Networks for 3 years. He has published seven papers in internationally recognized conferences/journals. His research interests include Multi-access Edge Computing, programmable radio access networks, AI-enabled autonomous networking, and distributed network management and orchestration.

Davit Harutyunyan received the bachelors and masters degrees (Hons.) in Telecommunication Engineering from the National Polytechnic University of Armenia in 2011 and 2015, respectively. H e also received the Ph.D. degree (Hons.) in Information and Communication Technology from the University of Trento in 2019. He was a Radio Access Network Optimization Engineer with Orange Armenia. Currently, he is an expert researcher in Wireless and Networked Systems Research Unit at FBK CREATE-NET. His main research interests include software-defined mobile networking, next-generation radio access networks, multi-access edge computing and virtualization technologies. He has published ten papers in internationally recognized journals/conferences. He was the recipient of the Best Student Paper Award of IEEE CNSM 2017 and IEEE NetSoft 2019.

Roberto Riggio is Head of the Wireless and Networked System Unit at FBK CREATE-NET. His research interests include software-defined mobile networks, network slicing, and distributed management and orchestration of network services. He has published more than 100 papers and has generated more than 3M Euro in competitive funding. He received several awards including the IEEE CNSM 2015 Best Paper Award. He serves in the TPC/OC of leading conferences in networking and is associate editor of several journals including the IEEE Transactions on Network and Service Management.