Performance Assessment of an ITU-T Compliant Machine Learning Enhancements for 5G RAN Network Slicing

Natalia Yarkina, Anna Gaydamaka, Dmitri Moltchanov, Yevgeni Koucheryavy

Abstract—Network slicing is a technique introduced by 3GPP to enable multi-tenant operation in 5G systems. However, the support of slicing at the air interface requires not only efficient optimization algorithms operating in real time but also its tight integration into the 5G control plane. In this paper, we first present a priority-based mechanism enabling defined performance isolation among slices competing for resources. Then, to speed up the resource arbitration process, we propose and compare several supervised machine learning (ML) techniques. We show how to embed the proposed approach into the ITU-T standardized ML architecture. The proposed ML enhancement is evaluated under realistic traffic conditions with respect to the performance criteria defined by GSMA while explicitly accounting for 5G millimeter wave channel conditions. Our results show that ML techniques are able to provide suitable approximations for the resource allocation process ensuring slice performance isolation, efficient resource use, and fairness. Among the considered algorithms, polynomial regressions show the best results outperforming the exact solution algorithm by 5–6 orders of magnitude in terms of execution time and both neural network and random forest algorithms in terms of accuracy (by 20–40 %), sensitiveness to workload variations and training sample size. Finally, ML algorithms are generally prone to service level agreements (SLA) violation under high load and time-varying channel conditions, implying that an SLA enforcement system is needed in ITU-T's 5G ML framework.

Index Terms—5G, network slicing, machine learning, radio access network, slice isolation, ITU-T.

1 Introduction

The introduction of the 5G cellular architecture not only drastically enhances the amount of resources at the air interface but enables flexibility of the end-to-end resource control and management [1], [2]. One of its advanced functionalities is network slicing providing the tools for efficient resource management including the radio access network (RAN) [3].

Following 3GPP [4], a network slice is a logical network that provides specific network capabilities and network characteristics. The specification also demands slice isolation, although defined in a broad sense encompassing multiple levels such as security, performance, etc. Providing performance isolation of slices along with efficient use of system resources and fairness of their allocation is a difficult task since these requirements are largely contradictory [5], [6]. The problem is even more challenging when slicing is extended to the RAN, where dynamic channel conditions need to be accounted for when designing isolation schemes.

To date, a number of algorithms have been proposed for network slicing in RAN with various performance isolation criteria taken into account. As most of those approaches formalize and solve an optimization problem, the solution complexity becomes a critical issue for practical implementation of the proposed algorithms. On top of this, many of the formulated slicing problems do not account for specifics of wireless propagation by abstracting the cell capacity. In dynamically changing wireless channel conditions, ensuring both isolation and fairness of resource allocation may lead to inability to timely redistribute resources among slices and flows that belong to different slices. As a result, lightweight approximations of exact solutions are of special importance for practical implementations.

Machine learning (ML) has recently become a viable alternative to conventional optimization and prediction techniques in various applied fields of science. Industry, transportation, healthcare, and many other fields utilize ML to solve a wide range of problems. There are numerous use cases of ML in communication networks [7], network automation and optimization [8], anomaly detection, network traffic prediction [9], [10], traffic optimization adjustment, etc. In the context of 5G systems, ITU-T has recently standardized a framework for ML integration by specifying the corresponding architecture and data handling [11], [12].

This study aims at improving practical applicability of an existing RAN slicing scheme by enhancing it with a lightweight ML-based approximator capable of providing a solution fast enough to follow the evolution of radio channel conditions. The investigated approach is applicable to other RAN slicing schemes that imply solving one or more optimization problems under very strict time constraints typical for RAN resource allocation.

In this paper, we assess the use of ML for RAN slicing within the ITU-T architecture framework. We first formalize a model of resource slicing in RAN aimed at fair priority-based isolation of slices. Then, we apply supervised ML techniques to reduce the solution complexity while accounting for specifics of wireless propagation and a realistic slice content composition. Particularly, we analyze performance

The authors are with the Department of Electrical Engineering and Communications, Tampere University, Tampere, Finland. Email: {natalia.yarkina, anna.gaydamaka, dmitri.moltchanov, evgeni.kucheryavy}@tuni.fi

of four candidate ML models – the linear regression, the polynomial regression, the random forest regressor, and the artificial neural network (ANN) – chosen primarily for their prediction and training speed. We consider their online and offline implementations and discuss how to integrate them into the 5G ML architecture standardized by ITU-T.

The main conclusions of our work are:

- *accuracy and implementation:* (i) the polynomial regressions of degrees 2 and 4 show the best results in the online learning setting outperforming other models in terms of execution time, accuracy, generalization capability, and training sample size, which makes them suitable for online implementation with frequently changing traffic distributions across slices; (ii) when trained offline and tested on simulation data, the models show an accuracy level inferior to the online training, however, for random forest regressors, this could be improved via larger training datasets and increased model complexity;
- *sensitiveness:* (i) ML algorithms are sensitive to the composition of slices and workload distribution across them, but not to the overall workload; (ii) prediction accuracy does not decrease with the number of slices, which makes ML enhancement particularly suitable for over 5–7 slices due to slow exact computation;
- *channel and traffic impairments:* ML algorithms are prone to service level agreements (SLA) violation under high load and time-varying channel conditions, implying that an SLA control system in real time is needed in the ML pipeline.

The rest of the paper is organized as follows. In Section 2 we discuss the ITU-T standardized framework for ML implementation in 5G networks and also briefly review the recent work related to ML enhancement of network slicing. Section 3 presents the system model and its components. In Section 4 we describe the adopted slicing scheme and provide an exact solution algorithm. Section 5 introduces the ML enhancement framework and techniques for speeding up resource arbitration. In Section 6 we present the adopted numerical evaluation scenarios and study stochastic process representing the cell capacity. Numerical results and their interpretation are provided in Section 7. Conclusions are drawn in the last section.

2 Background and Related Work

In this section, we discuss the use of ML in 5G systems including the ITU-T architectural framework for ML integration and applications of ML techniques for network slicing. We refer the reader to [13] for a survey of 5G network slicing enablers, architectures and deployment strategies, and to [5], [14] for recent reviews on RAN resource allocation to slices.

2.1 Machine Learning Integration in 5G Cellular

ML is defined in [11] as a process that enables computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed. Fig. 1 shows the high-level architecture for enabling ML in a

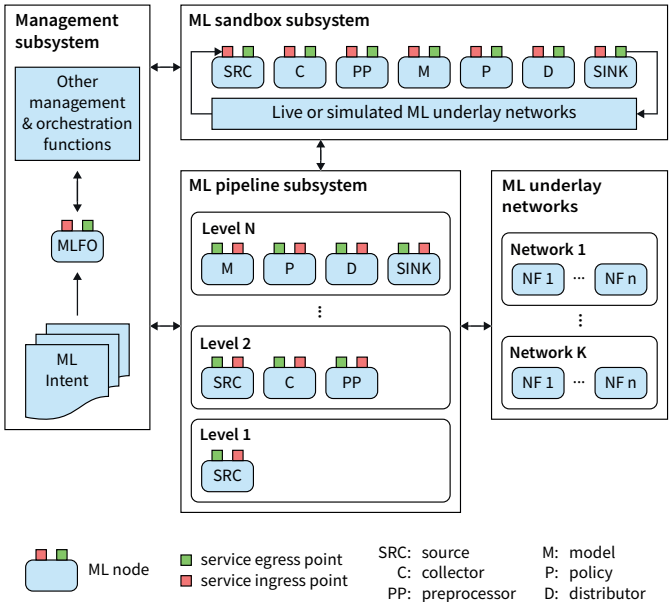


Fig. 1. ITU-T high-level architecture for ML integration (adapted from [11]).

telecommunication network which was proposed in ITU-T Y.3172 [11]. Its main components are the ML pipeline, the ML function orchestrator (MLFO), and the ML sandbox.

The ML pipeline is a set of logical nodes with specific functionalities. The nodes are combined to form an ML application. The source node (SRC) provides input data for the ML pipeline. The collector node (C) is responsible for collecting data from one or more source nodes. All data preprocessing, including data cleaning and aggregation, is performed by the preprocessor node (PP). One of the key nodes, the model node (M), is in charge of executing the chosen ML model. The output of the model can be revised by the policy node (P) and some rules can be applied to it. Thereafter the distributor node (D) manages the output and delivers it to one or more sink nodes (SINK). The sink node represents the target of the ML output where it takes action.

The ML pipeline can be overlaid on an existing network infrastructure with its nodes positioned and chained throughout several network levels (e.g., UE, the access network and the core network). The placement of ML functionalities is governed by such factors as the specifications of ML applications, their latency constraints and availability of data, but also performance and resource constraints of network functions (NF) and levels. The placement and chaining of the ML pipeline nodes are controlled by MLFO, a logical node that manages and orchestrates the nodes of ML pipelines. The input for MLFO is ML Intent, which represents a declarative description specifying an ML application. Based on this information and network conditions MLFO can control, arrange and change the nodes of ML pipelines. To implement an ML application, MLFO, in coordination with other management and orchestration functions, instantiates nodes of an ML pipeline with specific roles (e.g., SRC, C, M) and associates them to technology-specific NFs of the underlying network based on their capabilities and corresponding requirements of the ML application.

The ML sandbox is an isolated domain used for train-

ing, testing and evaluating ML pipelines before deploying them in a live network. The ML pipelines hosted here are separate, but the data can come from both simulated and live underlay networks. If any changes occur in ML Intent or new specifications are added, MLFO updates the ML pipeline nodes in the ML sandbox so that they correspond to the modified scenario.

2.2 Machine Learning for Network Slicing

Applications of ML for network slicing enhancement can be manifold [15], however, resource allocation among slices receives the most attention from researchers. Indeed, in the majority of policies proposed so far for network slicing in RAN, the resource shares allocated to slices/slice users are determined as a solution to a linear or non-linear optimization problem [5]. However, considering the numerous constraints and the dimension of the problem, obtaining such a solution fast enough for a real-time adaptive resource reallocation can be challenging. A possible way to tackle this issue is by using ML techniques.

The survey [15] discusses the automation of numerous network functions involved in control and management of slices. The authors provide a list of 5G network slicing scenarios and suggest several ML techniques that can be adopted to enhance various slicing-related tasks. The authors of [16] develop a three-stage hybrid learning algorithm to classify network traffic into three slice categories: eMBB, mMTC or URLLC. The classification is based on the data such as user device type, session duration, packet delay budget, etc. The study in [17] investigates how traffic of one slice affects traffic of another slice. The authors develop a data-driven ML-based slicing and allocation model which intelligently assigns and redistributes resources among network slices with respect to certain quality of service (QoS) parameters. Similar problems are addressed in [18], where the authors build an experimental prototype of the 5G network architecture and embed ML solutions to configure radio resources for network slices. Their results show that despite the growing computing resource utilization the throughput of the network was increased. The authors in [19] focus on the problem of resource allocations with changing channel characteristics. They suggest implementing ML to correctly model the wireless channel.

A number of deep reinforcement learning (DRL) solutions, namely [20], [21], [22], [23], have been proposed for network slicing in RAN as an alternative to explicit optimization-problem-based and algorithmic resource sharing schemes. The authors of [20] investigate the feasibility and efficiency of applying the DRL framework to resource allocation among slices and consider two scenarios: priority-based core network slicing and radio resource slicing. In the latter, a weighted sum of spectral efficiency and QoS in the three slices under study is adopted as the reward function for the decision process. The authors of [21] propose a combined solution including a deep recurrent neural network to predict traffic volume in large timescales and a reinforcement learning algorithm for performing the resource scheduling in small timescales. Here, the reward function aims at minimizing the resource consumption while guaranteeing a certain degree of slice performance isolation and

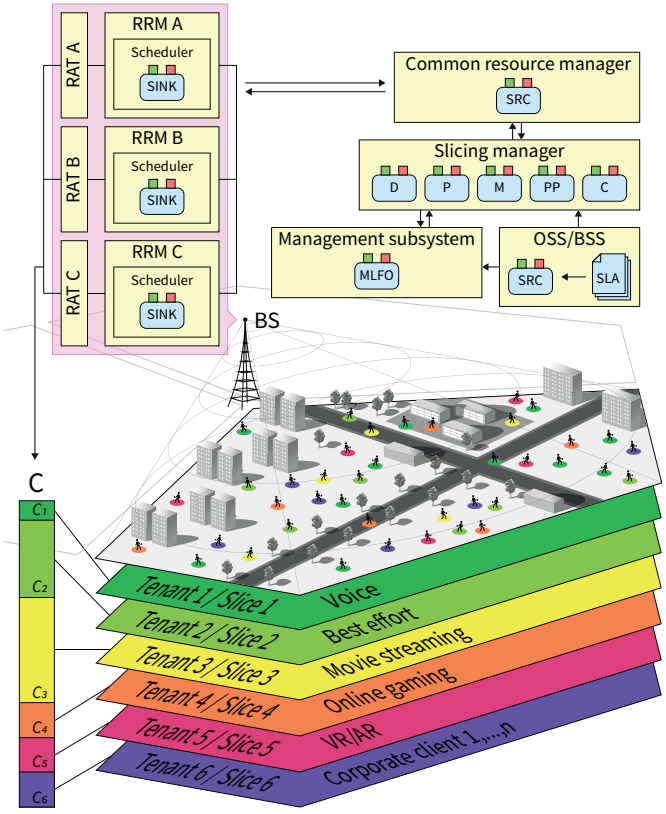


Fig. 2. The considered ML-enabled slicing system architecture.

includes a bonus for slice reconfiguration. In [22] the authors propose a system for dynamic reservation of unused resources in virtualized RAN based on DRL algorithms. They show that by tuning the objective function, which represents a weighted sum of average slice-specific QoS utilities and per slice resource utilizations, significant improvements in resource utilization can be achieved, however, no guidance is provided on the choice of these functions.

Although DRL is considered a promising approach to resource sharing in context of network slicing, it is characterized by high training complexity [20] and a certain arbitrariness in the choice of the reward function resulting in impeded tractability. Moreover, most studies consider workload, where there is no high competition for resources among slices and do not investigate the efficiency of the ML model under workload bursts, which might be compromised. The rationale behind our proposed framework is to combine the tractability of an explicit slicing policy covering all workload ranges with the time efficiency of the simplest supervised ML techniques, which can approximate the solution when its computation by the exact algorithm takes too long.

3 System Model

In this section, we introduce our system model and its components. We start with the overall system design and then proceed with the radio part providing an abstraction of resources at the air interface. Then, we specify the traffic process for each slice and introduce the slice performance isolation policy and our approach to its ML enhancement.

3.1 Base Station with ML-enhanced RAN Slicing

We study the downlink transmission of a 5G base station (BS) providing virtualization of radio access resources and network slicing (see Fig. 2). The uplink direction and the mixed uplink/downlink case can be addressed in a similar manner. The BS may have several radio access technologies (RAT A, RAT B, etc.), whose resources are controlled in the radio resource management (RRM) subsystems. The RRMs are collectively controlled and coordinated by a common resource manager (CoRM), which combines the data rates provided by each RAT into the aggregated time-varying BS capacity, $C(t) = c_A(t) + c_B(t) + \ldots, t \geq 0$, and manages its allocation to user sessions. The slicing manager is a separate entity responsible for dynamic distribution of the aggregated capacity, $C(t)$, among S instantiated network slices based upon the adopted slicing policy and demand. The resulting capacity allocation to slices, $C_1(t), C_2(t), \ldots, C_S(t)$, is communicated back to CoRM and translated to resource allocation constraints at each RAT scheduler. The considered network structure corresponds to a heterogeneous network of a single operator – the infrastructure provider (InP).

The slicing manager's operation is enhanced by ML. The ML pipeline implemented therein receives data characterizing the aggregated capacity and the slices' demand therefor from CoRM. The data is fed to an ML model which computes $C_i(t)$, $i = 1, \ldots, S$, and returns them to CoRM in terms of shares of the total capacity for further coordinated resource allocation among RATs. CoRM instructs the RAT-specific RRMs to provide appropriate capacity to UEs. The slicing manager also uses parameters from the SLAs between the InP and the slices' tenants, which are accessible through the SLA management functions in the Operations/Business Support System (OSS/BSS).

An MLFO supervises the ML pipeline's performance and the accuracy of the output. It calls for retraining if the accuracy is insufficient (e.g., due to a substantial change in demand) or the slicing parameters have changed (e.g., a new slice has been instantiated or SLA parameters modified). Retraining is performed in an ML sandbox (see Fig. 1), which has access to the data provided by CoRM and OSS/BSS.

3.2 Radio Specifics

Each considered RAT has an assigned frequency band. As opposed to many previous studies of RAT network slicing, we adopt a joint methodology and combine the slice-level resource allocation with an explicit account for wireless channel dynamics. To this aim, we utilize computer simulations to obtain the time-varying RAT capacity. A 3GPP-compliant radio channel modeling procedure accounting for propagation, antenna, user mobility, human-body blockage, and line-of-sight obstruction specifics is detailed in Section 6 while the radio part sub-models are introduced below.

3.2.1 Propagation Model

Throughout the paper we consider the most complex RAT, mmWave, as an example. We represent propagation losses using the Urban-Micro (UMi) Street-Canyon model.

Let $I_{\mathrm{LoS}} = 1$ if the UE is under the line-of-sight (LoS) and $I_{\mathrm{LoS}} = 0$ under non-line-of-sight (nLoS) conditions. Similarly, let $I_{\mathrm{nHB}} = 1$ if the UE is not blocked by human (nHB) and $I_{\mathrm{nHB}} = 0$ otherwise (i.e., human-blocked, HB). According to [24], the path loss for the frequency band 0.5–100 GHz can be expressed in dB as

$$\mathrm{PL}_{[\mathrm{dB}]}(d, I_{\mathrm{LoS}}, I_{\mathrm{nHB}}) = 10\alpha(I_{\mathrm{LoS}}) \log_{10} d + \beta(I_{\mathrm{nHB}}) + 20 \log_{10} f_c + \chi_{\sigma_{\mathrm{SF}}(I_{\mathrm{LoS}})}, \quad (1)$$

where d is the three-dimensional (3D) distance in meters between the NR BS and the UE, $\alpha(I_{\mathrm{LoS}})$ is a coefficient being 2.1 under the LoS and 3.19 under nLoS conditions, $\beta(I_{\mathrm{nHB}})$ is 32.4 dB when the UE is not blocked by human and 52.4 dB otherwise, f_c is the carrier frequency measured in GHz, and $\chi_{\sigma_{\mathrm{SF}}(I_{\mathrm{LoS}})}$ is the shadow fading in dB, which is normally distributed with zero mean and standard deviation $\sigma_{\mathrm{SF}}(I_{\mathrm{LoS}})$. Note that the value of $\sigma_{\mathrm{SF}}(I_{\mathrm{LoS}})$ also depends on I_{LoS} [24].

By converting the path loss (1) to linear scale we can write the received signal-to-interference-plus-noise ratio (SINR) as

$$\mathrm{SINR}(d, I_{\mathrm{LoS}}, I_{\mathrm{nHB}}) = \frac{P_{\mathrm{BS}} G_{\mathrm{BS}} G_{\mathrm{UE}}}{\mathrm{PL}(d, I_{\mathrm{LoS}}, I_{\mathrm{nHB}}) N_0 B_{\mathrm{PRB}} A}, \quad (2)$$

where P_{BS} is the emitted power, G_{BS} and G_{UE} are respectively the BS and UE antenna gains, N_0 is the thermal noise power spectral density, B_{PRB} is the size of the physical resource block (PRB), and A represents aggregated losses given, in decibels, by

$$A_{[\mathrm{dB}]} = M_{\mathrm{I}} + F_{\mathrm{N}} + L_{\mathrm{C}} \quad (3)$$

TABLE 1
Notation utilized in this paper

Notation	Description
A	Aggregated losses
B	Bandwidth
B_{PRB}	Size of the physical resource block
C	BS capacity
C_s	Capacity of slice s
$\mathcal{D}$	Labeled dataset
d	Distance between BS and UE
f_c	Operating frequency
$G_{\mathrm{BS/UE}}$	BS/UE antenna gain
$h_{\mathrm{B/BS/UE}}$	Height of Blocker/BS/UE height, $\Delta h = h_{\mathrm{BS}} - h_{\mathrm{UE}}$
I_{LoS}	Indicator of LoS blockage, 1 if LoS, 0 if nLoS
I_{nHB}	Indicator of HB blockage, 1 if not blocked, 0 otherwise
K	Dataset size
K_X	Number of antenna elements
N	Number of active users
N_0	Noise power spectral density
N_s	Number of ongoing sessions in slice s
N_s^{cont}	Contracted number of sessions in slice s
P_{BS}	Transmitting BS power
p_{out}	Outage probability
PL	Path loss
q_s	Probability for an arriving session to go to slice s
r_{BS}	BS service radius
R_s	User data rate in slice s
R_s^{min}	Minimum data rate per user in slice s
R_s^{max}	Maximum data rate per user in slice s
$\mathcal{S}$	Set of all instantiated slices, $S = \lvert\mathcal{S}\rvert$
SINR	Received signal-to-interference-plus-noise ratio
v	User speed
γ_s	Contracted capacity share of slice s
θ_s	Mean session duration in slice s
ν	Session arrival rate
σ_{SF}	Shadow fading standard deviation
τ_{RDM}	Mean Random Direction Mobility run time

with M_{I} being the interference margin, F_{N} the noise figure, and L_C the cable losses.

The cable losses L_{C} depend on the UE implementation and we assume $L_{\mathrm{C}} = 2$ dB, see Table 2 [25]. The noise figure, F_{N}, also relates to the UE implementation and characterizes the amount of noise generated by the device itself with no signal present. We also set it to a typical value, 7 dB, but it may vary from device to device in the range 2–10 dB [25]. Finally, interference is usually represented as a random variable depending on many factors including the type of deployment (random or cellular-like), operational frequency, deployment density, the use of resource blocks at neighboring cells, etc. However, interference models developed for mmWave systems have shown that the use of directional antennas greatly reduces interference as compared to microwave systems. This is why, and also to simplify the propagation model, we utilize the interference margin to capture inter-cell interference. Although we set the margin to 3 dB, one can utilize the models in [26], [27], [28] to estimate the mean interference in a specific setup and use it as M_I.

As the timescale of interest in this paper is not less than a few tens/hundreds of transmission time intervals (TTIs), we exclude the fast fading phenomena from consideration in (2).

3.2.2 Blockage Conditions

Following [24], for a user at a 2D distance d meters from the BS, the LoS probability is

$$\mathrm{P}\{I_{\mathrm{LoS}}{=}1|d\} = \begin{cases} 1, & d \leq 18, \\ 18d^{-1} + e^{-\frac{d}{36}}(1 - 18d^{-1}), & d > 18. \end{cases} \tag{4}$$

In our model the user's LoS/nLoS state, which is indicated by I_{LoS}, is chosen randomly according to (4) and remains unchanged for a time period exponentially distributed with mean τ_{LoS}. The latter is interpreted as the time to cross a building block at a pedestrian speed v.

An attenuation of 20 dB induced by human-body blockage [29] is reflected in the value of $\beta(I_{\mathrm{nHB}})$ in (1). Following [30], we assume that human blockers are represented by cylinders with a base radius of r_{B} and a height of h_{B} meters. Then, the human-body blockage probability is given by [30]

$$\mathrm{P}\{I_{\mathrm{nHB}}{=}0|d\} = 1 - e^{-2\zeta_{\mathrm{HB}} r_{\mathrm{B}} \left(\sqrt{d^2+\Delta h^2}\frac{h_{\mathrm{B}}-h_{\mathrm{UE}}}{\Delta h} + r_{\mathrm{B}}\right)}, \tag{5}$$

where ζ_{HB} is the density of blockers per square meter and $\Delta h = h_{\mathrm{BS}} - h_{\mathrm{UE}}$ with h_{BS} and h_{UE} being the BS and UE heights, respectively. Similarly, we sample the user's HB/nHB state by (5) and it does not change for a random time interval exponentially distributed with mean τ_{HB}.

3.2.3 Antenna Model

Linear antenna arrays are assumed at both UEs and the BS. To model the radiation patterns, similarly to [31], we utilize cone models with a constant gain over the main lobe. We denote by K_X, $X \in \{\mathrm{BS}, \mathrm{UE}\}$, the number of antenna elements and assume the distance between the neighboring elements to be $\lambda/2$, where λ is the wavelength. The phase excitation difference between the elements is assumed zero. Then, following [32], the half-power beamwidth (HPBW) of the main lobe for a symmetrical pattern is given by $2|\theta_m - \theta^{\pm}_{3\mathrm{dB}}|$, where $\theta_m = \pi/2$ is the array orientation and $\theta^{\pm}_{3\mathrm{dB}} = \arccos(\mp\frac{2.782}{K_X \pi})$ are the half-power points' angles. The mean gain over the main lobe can be obtained, for $X \in \{\mathrm{BS}, \mathrm{UE}\}$, as [32]

$$G_X = \frac{1}{\theta^{+}_{3\mathrm{dB}} - \theta^{-}_{3\mathrm{dB}}} \int_{\theta^{-}_{3\mathrm{dB}}}^{\theta^{+}_{3\mathrm{dB}}} \frac{\sin(\frac{K_X \pi}{2}\cos\theta)}{\sin(\frac{\pi}{2}\cos\theta)} d\theta. \tag{6}$$

3.2.4 Cell Size and User Mobility

It is assumed that, upon session arrival, the user is randomly placed in the cell coverage area according to the uniform distribution. The coverage area is specified by the BS service radius such that under the worst-case blockage conditions – nLoS and HB – a cell edge UE experiences an outage (i.e., a SINR below a threshold value $\mathrm{SINR}_{\mathrm{thre}}$) no more than fraction p_{out} of time. It can be obtained as follows

$$r_{\mathrm{BS}} = \sqrt{\left(\frac{P_{\mathrm{BS}} G_{\mathrm{BS}} G_{\mathrm{UE}}}{10^{\frac{\beta(0)}{10}} f_c^2 N_0 B_{\mathrm{PRB}} A M_{\mathrm{SF}} \mathrm{SINR}_{\mathrm{thre}}}\right)^{\frac{2}{\alpha(0)}} - \Delta h^2}, \tag{7}$$

where M_{SF} is the slow fading margin given by

$$M_{\mathrm{SF[dB]}} = \sqrt{2}\mathrm{erfc}^{-1}(2p_{\mathrm{out}})\sigma_{\mathrm{SF}}(0) \tag{8}$$

with $\mathrm{erfc}^{-1}(\cdot)$ denoting the inverse complementary error function, p_{out} is the outage probability at the cell edge coinciding with the fraction of time UE is in outage conditions.

We track only users having active sessions and stop tracking a user as soon as his or her session terminates. To represent user movement, we adopt the Random Direction Mobility (RDM) model [33]. Accordingly, each user selects a random direction from the interval $[0, 2\pi)$ and moves in this direction at a fixed constant speed v during an individual run time exponentially distributed with mean τ_{RDM}. As soon as this run time period elapses, the procedure for the user is repeated. Whenever the user reaches the cell's boundary, the movement direction is reflected.

The mobility of users is assumed homogeneous and independent of each other. The flow of users across the cell boundary is assumed stationary.

3.2.5 Cell Capacity

We approximate the total cell capacity $C(t)$ at time $t \geq 0$ by assuming equal bandwidth sharing between the N active users as follows

$$C(t) = \frac{B}{N} \sum_{i=1}^{N} \eta_i(t), \tag{9}$$

where B denotes the bandwidth and $\eta_i(t)$ is the spectral efficiency of UE i at time t. The user's spectral efficiency, $\eta_i(t)$, is obtained by mapping SINR (2) to the NR modulation and coding scheme (MCS) [34].

Note that (9) is essentially an approximation and builds upon two assumptions: (i) the whole set of available resources is utilized and (ii) resources are equally partitioned among all users. We then employ $C(t)$ computed by (9) in Section 6.1 to characterize the cell capacity's dynamics, which is further used in Section 7. In practice the exact cell capacity's behavior depends on the exact slice resource allocations determined by the employed slicing policy, as

well as on resource allocations between sessions within each slice, which in turn may depend on the tenants' proper policies. Thus, the adopted approach permits a decomposition of the overall problem into separate tasks.

3.3 Traffic and Slices

The BS serves heterogeneous traffic and the network slicing technique is employed to efficiently accommodate sessions with substantially different QoS requirements (voice, video streaming, gaming, etc.). We denote the set of all instantiated slices by $\mathcal{S}$, $|\mathcal{S}| = S$, and assume each slice intended for one type of service, which makes it homogeneous in terms of session characteristics and QoS parameters.

Since slices are service-specific, for each of them we can define a minimum data rate per user, $R_s^{\min} > 0$, needed to meet the QoS requirements of the service provided in the slice, $\mathbf{R}^{\min} = (R_s^{\min})_{s\in\mathcal{S}}$. It is assumed that a user cannot receive proper service if the data rate is below this value. Furthermore, following GSMA NG.116 [35], for each slice we can specify a maximum user data rate. It is denoted by $R_s^{\max} \geq R_s^{\min}$, $\mathbf{R}^{\max} = (R_s^{\max})_{s\in\mathcal{S}}$, and corresponds to such a value that allocating a data rate higher than this will not result in any gain in QoS or quality of experience (QoE) for the user.

Let N_s be the number of ongoing user sessions in slice s and denote the row vector containing the numbers of users in all slices by $\mathbf{N} = (N_s)_{s\in\mathcal{S}}$. Each user is assumed to have only one connection in only one slice. If a user has multiple connections, it is considered and served as multiple users, one per connection. We assume that users arrive into the system according to a Poisson process of rate ν, are directed to slice $s \in \mathcal{S}$ with probability q_s and leave the system upon session completion. Session durations in slice s are exponentially distributed with mean θ_s.

We assume that resources are shared among slices using the slicing scheme with equitable-priority-based performance isolation of slices proposed in [5]. Let $C_s \geq 0$ represent the capacity of slice $s \in \mathcal{S}$ and let it follow the number of users in the slice in the form $C_s = N_s R_s$, where R_s is the user data rate in slice $s \in \mathcal{S}$ to be determined by the slicing scheme. Note that R_s is the ensemble average user data rate in the slice, and the actual data rates perceived by slice users may differ depending on their channel conditions and the resource allocation policy applied by the slice's tenant. Let $\mathbf{R} = (R_s)_{s\in\mathcal{S}}$ be a column vector. Considering that C is the total capacity of the BS, the capacities of slices must satisfy $\sum_{s\in\mathcal{S}} C_s \leq C$.

For our system, we demand that

$$R_s^{\min} \leq R_s \leq R_s^{\min},\ s \in \mathcal{S}, \tag{10}$$

as long as the number of users in the slice, N_s, does not exceed a contracted number N_s^{cont}, $\mathbf{N}^{\text{cont}} = (N_s^{\text{cont}})_{s\in\mathcal{S}}$. Indeed, due to capacity limitations, slice performance isolation cannot be guaranteed for unrestricted traffic in all slices, so it is assumed that slice isolation is ensured as long as the number of users in the slice does not exceed its contracted threshold, i.e., $N_s \leq N_s^{\text{cont}}$. The InP thus guarantees performance isolation of slice s by providing it with at least a capacity of

$$C_s^{\min}(N_s) = \min\{N_s, N_s^{\text{cont}}\} R_s^{\min}. \tag{11}$$

Any remaining capacity is distributed among all slices on the basis of fairness, but so that $R_s \leq R_s^{\max}$, $s \in \mathcal{S}$. Note that we allow for overbooking, i.e., the sum of the contracted slice capacities, $\sum_{s\in\mathcal{S}} N_s^{\text{cont}} R_s^{\min}$, can be larger than C.

The considered slicing scheme provides a flexible and dynamic partitioning of the total BS capacity among slices based upon (i) the parameters $\mathbf{R}^{\min}$, $\mathbf{R}^{\max}$ and $\mathbf{N}^{\text{cont}}$, and (ii) the demand expressed in terms of the number of users $\mathbf{N}$. It is assumed that the parameters $R_s^{\min}$, $R_s^{\max}$ and N_s^{cont} are agreed upon between the InP and the slice s tenant and stated in the corresponding SLA, with N_s^{cont} set either directly or in the form of the contracted resource share

$$\gamma_s = \frac{N_s^{\text{cont}} R_s^{\min}}{C},\ s \in \mathcal{S}. \tag{12}$$

Flexibility is assured by the fact that when some slices do not use all their contracted capacity $N_s^{\text{cont}} R_s^{\min}$, the remaining capacity $(N_s^{\text{cont}} - N_s) R_s^{\min}$ becomes available to other slices if they need it. Thus, each slice has priority to its contracted capacity over other slices.

Computation of $\mathbf{R}$ is specifically discussed in Section 4 and involves solving a convex programming problem, which can prove computationally challenging under the time constraints characterizing radio resource scheduling. The proposed ML enhancement addresses this challenge by providing time-efficient approximations using supervised ML techniques.

3.4 ML Enhancement

The architecture proposed by ITU-T and described in Subsection 2.1 is generic enough to be adopted in a multitude of scenarios [36]. In this work, we propose to employ supervised ML to approximate $\mathbf{C} = (C_s)_{s\in\mathcal{S}}$ based upon a labeled sample obtained by applying the exact solution algorithm. Moreover, as the parameters $\mathbf{N}^{\text{cont}}$, $\mathbf{R}^{\min}$ and $\mathbf{R}^{\max}$ change at a much larger timescale than the demand $\mathbf{N}$, they can be assumed constant, thus restricting the system's variability.

3.4.1 *Online Learning*

Two options for ML enhancement are investigated: online and offline. The online learning setting implies training on live data and repeatedly switching between the training and prediction phases. Each time a slice is instantiated, removed, or modified, a training phase begins. Here the slice capacities $\mathbf{C}$ are computed using the exact algorithm. The observed system states given by $\mathbf{N}$ along with the exact solutions are collected into a training dataset. Once the training set is populated, the implemented ML model is trained and the process moves on to the prediction phase. Here the ML technique is used to predict $\mathbf{C}$ from $\mathbf{N}$. The accuracy is constantly monitored either through periodical comparison with the exact solution or by assessing relevant performance measures. Whenever the system detects insufficient accuracy due to a change in demand yielding population vectors $\mathbf{N}$ substantially differing from the training data, the process starts over from the training phase.

3.4.2 Offline Learning

Whereas in the online learning setting the model is trained for a specific, current range of workloads, in the offline scenario training and validation data are sampled from the uniform distribution on the feasible space of $\mathbf{N}$. Labels for the data are computed using the exact algorithm with the average BS capacity. As a result, the trained model must be suitable for any workload regime and no accuracy monitoring is needed. The model has to be retrained only upon changes in the parameters $\mathbf{N}^{\text{cont}}$, $\mathbf{R}^{\min}$ and $\mathbf{R}^{\max}$.

3.5 Performance Assessment

Four supervised ML models are investigated in the paper: a linear regression, a polynomial regression, a random forest regressor and an ANN. However, other ML techniques can also be employed in the proposed settings. Our goal is to evaluate and compare these ML algorithms in terms of approximation accuracy, generalization capability and time efficiency under the online and offline learning scenarios. We also assess the impact of channel variability on the performance of these techniques. Specific performance criteria will be discussed in Subsection 5.7.

The timescale of interest for resource allocation considered in this paper is at least a few tens/hundreds of TTIs. Practically, it corresponds to the resource reallocation procedure invoked at either (i) regular intervals or (ii) at time instants when the slice conditions (e.g., the number of active users) change.

4 Equitable-Priority-based Slicing Policy

In this section, we first formalize a model of multi-tenant resource sharing at the air interface aimed at fair priority-based isolation of slices. Then, we proceed to presenting the exact solution algorithm and motivating the need for ML approaches for the overall problem solution.

4.1 Resource Arbitration Scheme

Since the demand if given in terms of the number of users in slices, the state of the BS is described by vector $\mathbf{N} \in \Omega = \mathbb{N}^S$. We partition the state space as $\Omega = \Omega^{\max} \cup \Omega^{\text{opt}} \cup \Omega^{\text{cong}}$, where

$$\Omega^{\max} = \{\mathbf{N} \in \Omega : \mathbf{N}\mathbf{R}^{\max} \leq C\} \tag{13}$$

contains all states in which the available capacity suffice to allocate the corresponding maximums to all users,

$$\Omega^{\text{opt}} = \{\mathbf{N} \in \Omega : \mathbf{N}\mathbf{R}^{\min} \leq C < \mathbf{N}\mathbf{R}^{\max}\} \tag{14}$$

contains all states in which the maximums cannot be allocated to all users yet minimums can, and

$$\Omega^{\text{cong}} = \{\mathbf{N} \in \Omega : \mathbf{N}\mathbf{R}^{\min} > C\} \tag{15}$$

contains all states in which even the corresponding minimums cannot be allocated to all users.

The slicing scheme determines the slice capacities $C_s(\mathbf{N}) \in \mathbb{R}_+$, $s \in \mathcal{S}$, for each $\mathbf{N} \in \Omega$ so that $\sum_{s\in\mathcal{S}} C_s(\mathbf{N}) \leq C$. For $\mathbf{N} \in \Omega^{\max}$, we can assign the maximum data rate to all users in all slices, which yields

$$C_s(\mathbf{N}) = N_s R_s^{\max},\ s \in \mathcal{S}. \tag{16}$$

For $\mathbf{N} \in \Omega^{\text{opt}}$, we seek to allocate resources in a way to (i) satisfy the minimum and maximum constraints (10), (ii) to make use of the whole available capacity and (iii) to provide max–min fairness to users taking account of whether the contracted number of users is exceeded and by how much. Such an allocation, for $\mathbf{N} \in \Omega^{\text{opt}}$, can be found as a solution to the optimization problem [5]

$$\text{SLICE}_{\mathbf{N},C}(\mathbf{R}^{\min}, \mathbf{R}^{\max}, \mathbf{N}^{\text{cont}})$$

$$\text{maximize } U(\mathbf{R}) = \sum_{s\in\mathcal{S}} W_s(N_s) N_s \ln(R_s) \tag{17}$$

$$\text{subject to } \mathbf{N}\mathbf{R} = C \tag{18}$$

$$\text{over } \mathbf{R} \in \mathbb{R}_+^S : R_s^{\min} \leq R_s \leq R_s^{\max} \tag{19}$$

with the weight functions defined as

$$W_s(N_s) = \begin{cases} 1, & N_s \leq N_s^{\text{cont}}, \\ N_s^{\text{cont}}/N_s, & N_s > N_s^{\text{cont}}. \end{cases} \tag{20}$$

The target function $U(\mathbf{R})$ in (17) is a utility function of the log type proposed for proportionally fair resource sharing in [37], which in our case coincides with the max–min fairness [38]. The weight functions (20) ensure a max–min fair resource allocation to users as long as their number in the corresponding slices does not exceed the contracted quantity, and penalize the "violating" slices by decreasing their weights. The constraint (18) ensures not only that the total allocation does not exceed the available capacity C, but also that all available capacity is allocated. Finally, the box constraints (19) ensure that the minimum and maximum data-rate requirements in slices are satisfied.

We now extend our policy to congestion states, $\mathbf{N} \in \Omega^{\text{cong}}$. Denote $N_s^{\min}(N_s) = \min\{N_s, N_s^{\text{cont}}\}$ and $\mathbf{N}^{\min} = \left(N_s^{\min}\right)_{s\in\mathcal{S}}$. Now, if $\mathbf{N}^{\min}\mathbf{R}^{\min} \geq C$ then we set

$$C_s(\mathbf{N}) = \frac{N_s^{\min} R_s^{\min}}{\mathbf{N}^{\min}\mathbf{R}^{\min}} C. \tag{21}$$

If, conversely, $\mathbf{N}^{\min}\mathbf{R}^{\min} < C$, then we can allocate the due capacity $\mathbf{N}^{\min}\mathbf{R}^{\min}$ and then need to distribute the remaining capacity $C - \mathbf{N}^{\min}\mathbf{R}^{\min}$ to $\mathbf{N} - \mathbf{N}^{\min}$ users. We will do this proportionally to the requested capacity as well, which yields

$$C_s(\mathbf{N}) = N_s^{\min} R_s^{\min} + \frac{(N_s - N_s^{\min}) R_s^{\min}}{(\mathbf{N} - \mathbf{N}^{\min})\mathbf{R}^{\min}} (C - \mathbf{N}^{\min}\mathbf{R}^{\min}). \tag{22}$$

The main features of the scheme, namely fairness and slice performance isolation, are illustrated numerically in Subsection 7.1

4.2 Exact Solution of $\text{SLICE}_{\mathbf{N},C}(\mathbf{R}^{\min}, \mathbf{R}^{\max}, \mathbf{N}^{\text{cont}})$

Since the objective function in $\text{SLICE}_{\mathbf{N},C}(\cdot)$ is differentiable and strictly concave and the feasible region is compact and convex, there is a unique maximum for $U(\mathbf{R})$ in the feasible region, which can be found by Lagrangian methods. For finding the exact solution of the problem we propose to employ Algorithm 1. It uses a recursive function, *FUNC*$(\mathbf{N}, \mathbf{R}, C, s)$, which populates the set of solution candidates, $\mathcal{R}$, by considering all possible combinations of active constraints (19).

Algorithm 1: Exact Solution of $\text{SLICE}_{\mathbf{N},C}(\mathbf{R}^{\min}, \mathbf{R}^{\max}, \mathbf{N}^{\text{cont}})$

Input: state $\mathbf{N} \in \Omega^{\text{opt}}$, parameters $\mathbf{R}^{\min}$, $\mathbf{R}^{\max}$, $\mathbf{N}^{\text{cont}}$ and C, the weights $\mathbf{W}$ obtained by (20)
Output: $\mathbf{R}$ solving (17)–(19)
Function `FUNC` $(\mathbf{N}^{\text{prev}}, \mathbf{R}^{\text{prev}}, C^{\text{prev}}, \hat{s})$**:**
  if $\mathbf{N}^{\text{prev}}\mathbf{R}^{\min} \leq C^{\text{prev}} \leq \mathbf{N}^{\text{prev}}\mathbf{R}^{\max}$ **then**
    $\mathbf{R} := \mathbf{R}^{\text{prev}}$
    $\mathbf{R}^{\text{SP}} := \frac{C^{\text{prev}}}{\mathbf{N}^{\text{prev}}\mathbf{W}}\mathbf{W}$ `// stationary point`
    if $R_s^{\min} \leq R_s^{\text{SP}} \leq R_s^{\max}\ \forall s \in \mathcal{S}$ *such that* $N_s^{\text{prev}} > 0$ **then**
      for *such* $s \in \mathcal{S}$ *that* $N_s^{\text{prev}} > 0$ **do**
        $R_s := R_s^{\text{SP}}$
      $\mathcal{R} := \mathcal{R} \bigcup \{\mathbf{R}\}$ `// add candidate`
      if $\hat{s} = 1$ **then**
        return
    else
      for *such* $s = \hat{s}, ..., S$ *that* $N_s^{\text{prev}} > 0$ **do**
        $\mathbf{N}^{\text{next}} := \mathbf{N}^{\text{prev}}$
        $\mathbf{R}^{\text{next}} := \mathbf{R}^{\text{prev}}$
        $N_s^{\text{next}} := 0$
        for $R_s^{\text{bnd}} \in \{R_s^{\min}, R_s^{\max}\}$ **do**
          $R_s^{\text{next}} := R_s^{\text{bnd}}$
          $C^{\text{next}} := C^{\text{prev}} - N_s^{\text{prev}} R_s^{\text{bnd}}$
          `FUNC` $(\mathbf{N}^{\text{next}}, \mathbf{R}^{\text{next}}, C^{\text{next}}, s+1)$

$\mathcal{R} := \emptyset$ `// set of candidate solutions`
`FUNC` $(\mathbf{N}, \mathbf{R}^{\max}, C, 1)$ `// populate` $\mathcal{R}$
$\mathbf{R} := \arg\max_{\hat{\mathbf{R}} \in \mathcal{R}} U(\hat{\mathbf{R}})$

The algorithm operates as follows. The unique solution to the problem (17)–(18), i.e., with the box constraints (19) lifted, can be easily found as

$$R_s^{\text{SP}} = \frac{W_s C}{\mathbf{N}\mathbf{W}},\ s \in \mathcal{S}. \tag{23}$$

If the stationary point, $\mathbf{R}^{\text{SP}}$, satisfies (19), then it is the sought-for optimum and $|\mathcal{R}| = 1$. If, conversely, $\mathbf{R}^{\text{SP}}$ is not feasible then *FUNC*$(\cdot)$ is run recursively with one additional constraint – either $R_s = R_s^{\max}$ or $R_s = R_s^{\min}$ – activated at each call for all $s \in \mathcal{S}$ corresponding to non-zero N_s. If, for instance, the boundary $R_{s^*} = R_{s^*}^{\max}$ is activated, then R_{s^*} is set to $R_{s^*}^{\max}$ in the solution candidate while its other entries are searched for as the solution to the problem under study with $C - N_{s^*}R_{s^*}^{\max}$ in place of C and N_{s^*} set to zero, hence the recursion. Once all possible combinations of active constraints have been considered and $\mathcal{R}$ populated, the vector maximizing the objective function (17) is chosen among the members of $\mathcal{R}$.

Note that whenever (23) does not provide a feasible solution, the time complexity of Algorithms 1 is exponential in S. The problem was tackled by iterative methods, namely the Gradient Projection method, in [5], however, it implied matrix inversion, which brings its complexity to $\mathcal{O}(S^4)$ in the worst case. Under high traffic conditions, when the number of sessions in slices may change on sub-second timescales, and when the number of slices is rather high this could be problematic for implementation. For this reason, we need faster algorithms that can be found in the ML field. In the next section, we consider several such approximations to speed up the resource arbitration process.

5 Supervised ML Techniques

In this section, we discuss the motivation and specifics of using supervised ML to enhance the considered resource arbitration procedure. Then, we formulate the corresponding ML regression problem and introduce the supervised ML techniques under analysis. Finally, we present the performance criteria that we use for comparing the techniques.

5.1 ML Enhancement Motivation and Specifics

Under high traffic conditions when the number of sessions in slices may change on sub-second timescales and when the number of slices is relatively large, obtaining the slice allocation $\mathbf{C} = (C_s)_{s\in\mathcal{S}}$ via the procedure described in Section 4 could be problematic for $\mathbf{N} \in \Omega^{\text{opt}}$ as it implies solving the optimization problem $\text{SLICE}_{\mathbf{N},C}(\cdot)$, whose exact solution is of exponential time complexity. To quicken the procedure for this range of system states, we propose employing time-efficient supervised ML techniques permitting to approximate the solution based upon a number of exact solution samples.

Recall that the $\text{SLICE}_{\mathbf{N},C}(\mathbf{R}^{\min}, \mathbf{R}^{\max}, \mathbf{N}^{\text{cont}})$ parameters are of different nature. The SLA thresholds, $\mathbf{R}^{\min}$, $\mathbf{R}^{\max}$ and $\mathbf{N}^{\text{cont}}$, change far less frequently than the demand $\mathbf{N}$ (e.g., when a slice is added or removed). We can hence assume them constant during prolonged periods of time and retrain the model each time they change. The BS capacity C, on the other hand, may vary much more frequently than $\mathbf{N}$, but, as it will be shown in the next section, remains concentrated around its mean value, which we denote by $\bar{C}$. We can thus assume that its fluctuations merely induce noise in sample labels and not consider it as an explicit parameter.

Finally, although $\text{SLICE}_{\mathbf{N},C}(\cdot)$ yields a solution in the form of data rates $\mathbf{R}$, we formulate the ML techniques for predicting slice capacities $\mathbf{C}$ and, consequently, use as sample labels $\mathbf{C} = \mathbf{N} \circ \mathbf{R}^T$, where $\circ$ denotes component-wise multiplication. This is due to the fact that C_s are much smoother mapping functions of $\mathbf{N}$ compared to R_s, which tremendously improves regression accuracy.

5.2 ML Problem Formulation

The above discussion brings us to the following ML problem formulation. Assume $\mathbf{R}^{\min}$, $\mathbf{R}^{\max}$, $\mathbf{N}^{\text{cont}}$ and $\bar{C}$ constant and consider training data

$$\mathcal{D} = \{(\mathbf{N}^{(1)}, \mathbf{C}^{(1)}), (\mathbf{N}^{(2)}, \mathbf{C}^{(2)}), \ldots, (\mathbf{N}^{(K)}, \mathbf{C}^{(K)})\}, \tag{24}$$

such that, for any $k = 1, \ldots, K$, sample $\mathbf{N}^{(k)}$ satisfies (14), i.e.

$$\mathbf{N}^{(k)}\mathbf{R}^{\min} \leq \bar{C} + \xi_k < \mathbf{N}^{(k)}\mathbf{R}^{\max}, \tag{25}$$

while $\mathbf{C}^{(k)} = \mathbf{N}^{(k)} \circ (\mathbf{R}^{(k)})^T$, where $\mathbf{R}^{(k)}$ is the solution to $\text{SLICE}_{\mathbf{N},C}(\cdot)$ with $\mathbf{N} = \mathbf{N}^{(k)}$ and $C = \bar{C} + \xi_k$. Quantities ξ_k, $k = 1, \ldots, K$, are realizations of a random variable ξ representing random deviation of the BS capacity from its

mean. Now, our task is to use $\mathcal{D}$ to build such a vector function $f(\mathbf{N}) = (f_s(\mathbf{N}))_{s\in\mathcal{S}}$ that $\widetilde{\mathbf{C}} = f(\mathbf{N})$ represents a suitable approximation for a slice allocation obtained from solving $\mathrm{SLICE}_{\mathbf{N},C}(\cdot)$ for any $\mathbf{N}$ satisfying (25).

For training the models we adopt the common approach and rely on the quadratic loss function (see, e.g., [39]), which in our case takes the form

$$\mathrm{MSE}(f,\mathcal{D}) = \frac{1}{KS}\sum_{k=1}^{K}\sum_{s=1}^{S}\left(f_s(\mathbf{N}^{(k)}) - C_s^{(k)}\right)^2. \quad (26)$$

Then, for each type of ML algorithm, the function f that minimizes the loss (26) is found and evaluated. In what follows, having training and execution complexity in mind, we specifically concentrate on two simple approaches (the linear and polynomial regressions), one with medium complexity (the random forest regressor) and the most complex one – ANN.

5.3 Linear Regression

The simplest technique we use is linear regression. Here, we predict the vector of slice capacities in state $\mathbf{N}$ as

$$\widetilde{\mathbf{C}} = \mathbf{x}\mathbf{B} = (1, N_1, \ldots, N_S)\mathbf{B}, \quad (27)$$

with the regression coefficients, $\mathbf{B}$, to be determined from the training data $\mathcal{D}$ to minimize (26). The matrix of regression coefficients can then be computed as [39]

$$\mathbf{B} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}, \quad (28)$$

where $\mathbf{X}$ has the form

$$\mathbf{X} = \begin{bmatrix} 1 & N_1^{(1)} & \ldots & N_S^{(1)} \\ 1 & N_1^{(2)} & \ldots & N_S^{(2)} \\ \vdots & & \ddots & \\ 1 & N_1^{(K)} & \ldots & N_S^{(K)} \end{bmatrix}, \quad (29)$$

and $\mathbf{Y} = [\mathbf{y}_1 \ \ldots \ \mathbf{y}_S]$ with column vectors $\mathbf{y}_s = (C_s^{(k)})_{k=\overline{1,K}}$.

The time cost to train and to query a linear regression model is, respectively, $\mathcal{O}(S^2K + S^3)$ and $\mathcal{O}(S)$ [39]. In our case, we have S such model, one for each C_s.

5.4 Polynomial Regression

In a polynomial regression of degree 2 the prediction is computed by (27) with vector $\mathbf{x}$ appended on the right with entries N_iN_j for all $i,j = 1,\ldots,S$ such that $i \le j$. The matrix of regression coefficients is obtained by (28) in which matrix $\mathbf{X}$ is appended on the right with respective columns $(N_i^{(k)}N_j^{(k)})_{k=\overline{1,K}}$ for all $i,j = 1,\ldots,S$ such that $i \le j$.

A polynomial regression of degree $m > 2$ is constructed from a regression of degree $m-1$ by the same procedure. Namely, the prediction is obtained by (27) in which vector $\mathbf{x}$ of a regression of degree $m-1$ is appended on the right with products of the form $N_1^{i_1} \times \cdots \times N_S^{i_S}$ for all $i_s = 0,1,2,\ldots$ such that $\sum_{s=1}^{S} i_s = m$. Matrix $\mathbf{X}$ in (28) is appended by respective columns of the form $((N_1^{(k)})^{i_1} \times \cdots \times (N_S^{(k)})^{i_S})_{k=\overline{1,K}}$ for all $i_s = 0,1,2,\ldots$ such that $\sum_{s=1}^{S} i_s = m$. Obviously, the linear regression is polynomial of degree 1.

5.5 Random Forest Regressor

The random forest algorithm constructs multiple decision trees on various sub-samples of the dataset and, in the case of the regressor, returns the average prediction of the individual trees. A decision tree is built by recursively partitioning the feature space so that the samples with the same or similar labels are grouped.

More specifically, let $\mathcal{D}_m$ be the training data at tree node m, $|\mathcal{D}_m| = K_m$. For each candidate split $\theta = (s, n_m)$ consisting of a feature s and its threshold value n_m, the data $\mathcal{D}_m$ is partitioned into $\mathcal{D}_m^-(\theta) = \{(\mathbf{N},\mathbf{C}) \in \mathcal{D}_m : N_s \le n_m\}$ and $\mathcal{D}_m^+(\theta) = \mathcal{D}_m \setminus \mathcal{D}_m^-(\theta)$. The quality of a candidate split of node m is computed using a loss function $L(\cdot)$ as [40]

$$g(\mathcal{D}_m,\theta) = \frac{K_m^-}{K_m}L(\mathcal{D}_m^-(\theta)) + \frac{K_m^+}{K_m}L(\mathcal{D}_m^+(\theta)). \quad (30)$$

The commonly used loss function for regression problems is, again, the mean squared error (MSE) [40], defined as

$$L(\mathcal{D}_m) = \frac{1}{K_m S}\sum_{(\mathbf{N},\mathbf{C})\in\mathcal{D}_m}\sum_{s=1}^{S}(C_s - \bar{C}_{m,s})^2 \quad (31)$$

with $\bar{C}_{m,s} = \frac{1}{K_m}\sum_{(\mathbf{N},\mathbf{C})\in\mathcal{D}_m} C_s$. The split minimizing (30), say θ^*, is then applied, and the algorithm recurses for subsets $\mathcal{D}_m^-(\theta^*)$ and $\mathcal{D}_m^+(\theta^*)$ until a predefined maximum depth is reached or $K_m = 1$. To predict the label of a sample $\mathbf{N}$ the algorithm starts at the root node of the decision tree and moves down the tree until a leaf is found. The sample is then associated with the label of this leaf.

According to [40], in general, the run time cost to construct and to query a balanced binary tree is, respectively, $\mathcal{O}(KS\log K)$ and $\mathcal{O}(\log K)$. The time complexity of a random forest with T balanced trees is $\mathcal{O}(TKS\log K)$ to train and $\mathcal{O}(T\log K)$ to query, attaining respectively $\mathcal{O}(TK^2S)$ and $\mathcal{O}(TK)$ in the worst case.

5.6 Artificial Neural Network

The last ML technique investigated is a fully connected ANN with M ReLU-activated hidden layers of size J_m, $m = 1,\ldots,M$, and a linear activation output. More specifically, the approximation is computed as

$$\begin{aligned} &\widetilde{\mathbf{C}} = (1, h_1^{(M)}, \ldots, h_{J_M}^{(M)})\mathbf{B}^{(M)}, \\ &\hat{\mathbf{h}}^{(m)} = (1, h_1^{(m-1)}, \ldots, h_{J_{m-1}}^{(m-1)})\mathbf{B}^{(m)},\ m = 2,\ldots,M, \\ &\hat{\mathbf{h}}^{(1)} = (1, N_1, \ldots, N_S)\mathbf{B}^{(0)}, \\ &h_j^{(m)} = \max\{0, \hat{h}_j^{(m)}\},\ j = 1,\ldots,J_m,\ m = 1,\ldots,M. \end{aligned} \quad (32)$$

The matrices $\mathbf{B}^{(m)}$, $m = 0,\ldots,M$, are obtained from $\mathcal{D}$ via backpropagation using the sum quadratic loss.

5.7 Performance Criteria

We evaluate and compare the considered ML techniques in terms of accuracy and performance within the settings described in Subsection 3.4. The predictive accuracy of an ML model f in a labeled dataset $\mathcal{D}$ of size K is assessed via

- the root mean square error

$$\mathrm{RMSE}(f,\mathcal{D}) = \sqrt{\mathrm{MSE}(f,\mathcal{D})}, \quad (33)$$

TABLE 2
Simulation parameters

Notation	Value	Description
B	400 MHz	Bandwidth
B_{PRB}	1.44 MHz	PRB size
f_c	28 GHz	Operating frequency
F_{N}	7 dB	Noise figure
G_{BS}	11.57 dBm	BS antenna gain
G_{UE}	5.57 dBm	UE antenna gain
h_{B}	1.7 m	Blocker height
h_{BS}	10 m	BS height
h_{UE}	1.5 m	UE height
L_{C}	2 dB	Cable losses
M_{I}	3 dB	Interference margin
N_0	– 174 dBm/Hz	Noise power spectral density
P_{BS}	24 dBm	Transmitting BS power
p_{out}	0.05	Outage probability
$\mathrm{SINR}_{\mathrm{thre}}$	– 9.478 dB	SINR threshold
T	10^4 s	Simulation length
Δt	1 s	Channel sampling interval
v	1.5 m/s	User speed
$\sigma_{\mathrm{SF}}(0)$	8.2 dB	nLoS shadow fading STD
$\sigma_{\mathrm{SF}}(1)$	4 dB	LoS shadow fading STD
τ_{RDM}	30 s	Mean run time in the RDM model
τ_{LoS}	30 s	Mean LoS/nLoS state time
τ_{HB}	3 s	Mean HB/nHB state time

- the mean absolute error

$$\mathrm{MAE}(f,\mathcal{D}) = \frac{1}{KS}\sum_{k=1}^{K}\sum_{s=1}^{S}\left|f_s(\mathbf{N}^{(k)}) - C_s^{(k)}\right|, \quad (34)$$

- the maximum residual error

$$\mathrm{MaxE}(f,\mathcal{D}) = \max_{k,s}\left|f_s(\mathbf{N}^{(k)}) - C_s^{(k)}\right|. \quad (35)$$

To evaluate performance of the ML techniques, we consider their time efficiency and the capability to satisfy the data rate constraints. The former is assessed via: (i) the prediction time and (ii) the sample dataset size needed for training, which is particularly relevant in the online learning setting.

The capability of the approximations to satisfy the constraints is estimated using the probability that the approximate resource allocation to slices results in SLA violation for capacity $\bar{C}$, that is,

$$V(f,\mathcal{D}) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\{\exists s\in\mathcal{S}: f_s\left(\mathbf{N}^{(k)}\right)\frac{\bar{C}}{\sum_{s=1}^{S} f_s(\mathbf{N}^{(k)})} < C_s^{\min}(N_s^{(k)})\}. \quad (36)$$

6 Evaluation Scenarios

In this section, we detail our simulation approach for estimating the total cell capacity and specify representative traffic characteristics. The resulting scenarios are then utilized to numerically assess performance of the ML techniques discussed in the previous section.

6.1 Dynamic Resource Characterization

We first determine the statistical characteristics of the stochastic process representing the cell capacity for different fixed numbers of users N. With each user we associate three independent Poisson processes (PP) of events. The first, of rate τ_{RDM}^{-1}, yields a sequence of times at which a new movement direction is selected randomly from the interval $[0, 2\pi)$. The second, of rate τ_{LoS}^{-1}, provides a sequence of times at which the user's LoS/nLoS state is selected according to the probability (4) in the current position of the user, and a new value for the shadow fading factor, $\chi_{\sigma_{\mathrm{SF}}[\mathrm{dB}]} \sim N(0, \sigma_{\mathrm{SF}}(I_{\mathrm{LoS}}))$, is sampled with the standard deviation corresponding to the selected LoS/nLoS state. The selected LoS/nLoS state and the shadow fading factor remain unchanged until the next event of this PP. Finally, the third PP is of rate τ_{HB}^{-1} and yields a sequence of times at which the user's HB/nHB state is selected according to probability (5).

Simulation time is divided into intervals of length Δt. At the beginning of each interval $[t_j, t_j + \Delta t)$, $j = 1, \ldots, T$, $t_1 = 0$, the new coordinates of user $i = 1, \ldots, N$ are calculated according to the RDM model as described in Subsection 3.2.4. If a user reaches the boundary of the BS service area defined by (7), its trajectory is reflected. At each time t_j, $j = 1, \ldots, T$, from the coordinates of each user i we calculate its 2D distance to the BS, $r_{i,j}$, and compute by (2) the user's SINR $\left(\sqrt{r_{i,j}^2 + \Delta h^2}, I_{\mathrm{LoS}}, I_{\mathrm{nHB}}\right)$. Then, the user's spectral efficiency, $\eta_i(t)$, is obtained from the NR MCS to SINR mapping [34]. The total BS capacity for $t \in [t_j, t_j + \Delta t)$ is approximated as $C(t) = C(t_j)$ by (9).

The system parameters utilized for cell capacity characterization are provided in Table 2. In this work, we deal with a relatively long-term resource allocation spanning the time of a few tens/hundreds of TTIs (the assumed TTI is the mmWave NR subframe corresponding to the numerology $\mu = 3$, i.e., 1 ms). Because we are concerned with the approximate total cell capacity obtained by averaging-out users' spectral efficiencies, the channel sampling interval, Δt, is directly set to a rather large value of 1 s. If, however, individual users' data rates are of interest, then one should track SINR at a much greater time resolution, e.g., 1 ms.

Fig. 3 shows the time series of the total cell capacity while Fig. 4 reports the cumulative distribution function (CDF), the probability density function (pdf) and the autocorrelation function (ACF) of $C(t)$ for 30, 40, 50 and 60 UEs. The data in Fig. 4 show that, regardless of the number of users, the distribution is close to log-normal reflecting the effect of a power-decaying propagation model.

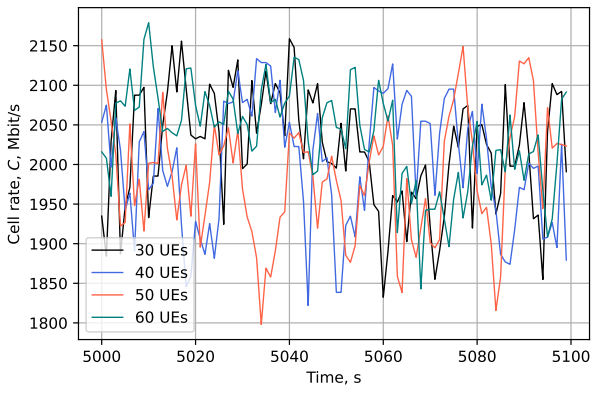
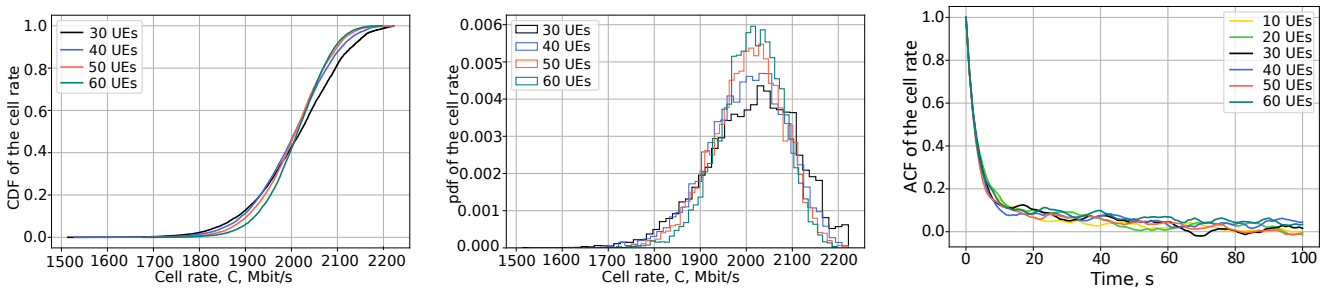


Fig. 3. Time series of the cell capacity.



(a) CDF of the cell capacity (b) pdf of the cell capacity (c) ACF of the cell capacity process

Fig. 4. Statistical characteristics of the cell capacity process $\{C(t), t \geq 0\}$.

Furthermore, for all the considered numbers of UEs in the system, the average total cell capacity remains unchanged at $\bar{C} \approx 2000$ Mbps.

As one may observe in Fig. 4(a) and 4(b), although the variance of the cell rate decreases with more UEs in the system, it still remains relatively large even for 60 UEs. This implies that the constant cell rate assumption, which is often made for simplicity reasons in slicing algorithms, may naturally require timely recalculation of the slice shares. The latter places strict constraints on the algorithms execution time, thus requiring lightweight ML solutions irrespective of the implementation type (online or offline). As further shown in Fig. 4(c), although the memory of the cell rate process is rather short, it remains very high over 2–4 s intervals providing a lower bound on the re-slicing frequency.

6.2 Traffic Characterization

A realistic assessment of traffic conditions is the key to creating network slicing models meeting modern requirements. To offer an insight into a typical traffic composition and its fluctuations, Table 3 provides traffic shares of the most significant application categories at different periods of the day based upon statistics collected in North America in 2021 and summarized in [41]. Furthermore, according to the same report, in the afternoon, the total traffic is about 20 % higher than during the morning hours. Also, video streaming traffic clearly predominates in downlink where its average share attains 48.9 %.

Using these data as a starting point and taking into account the projected growth in cloud gaming and virtual and augmented reality (VR/AR) applications [42], for our analysis we consider a composition of slices with the SLA

TABLE 3
Mobile internet traffic shares, %

Traffic Category	Morning	Afternoon	Evening
Video Streaming	19.2	28.6	32.5
Social Networking	29.0	27.0	26.0
Web Browsing	25.4	22.8	20.8
Gaming	0.5	1.7	1.3
Messaging	6.3	5.8	5.7
Marketplace	2.5	1.8	1.7
File Sharing	0.2	0.3	0.1
Cloud	9.2	5.6	5.5
VPN and Security	3.4	3.4	3.2
Audio	4.3	3.0	3.0

TABLE 4
SLA parameters of slices

Slice type	s	$R^{\min}_{s[\text{Mbps}]}$	$R^{\max}_{s[\text{Mbps}]}$	$\gamma_{s[\%]}$
Voice	1	0.1	1	4.5 fixed
Best Effort	2	1	C	2
Video Streaming	3	3	25	10.8
Gaming	4	10	50	8
VR/AR	5	20	100	14
Corporate	$6, \ldots, S$	5	50	4.75

and workload parameter values provided in Tables 4 and 5 respectively. Recall that the considered SLA parameters are the minimum and maximum user data rates $R^{\min}_s$ and $R^{\max}_s$, and the contracted slice resource share γ_s (12), to which the slice's sessions have priority over other slices. The workload parameters include the mean slice session durations θ_s, the user arrival rate in the cell ν, and the probability q_s for an arriving user to be served by slice $s \in \mathcal{S}$. The interarrival times and durations of sessions are assumed exponentially distributed.

The voice slice in our list is intended for classical cellular telephony services. Unlike all other slices, we assume it non-elastic in the sense that its size does not scale in and out with the demand due to the low data-rate requirement of a voice session; its capacity share is thus fixed. The best effort slice serves the traffic of social networking, web browsing, messaging, file sharing and other services that usually do not require strict QoS guarantees. The video streaming slice receives traffic of video streaming applications that provide quality guarantees to users. Gaming and VR/AR services are the most resource-demanding with strict latency requirements [43], and therefore have dedicated slices. Finally, corporate slices are assumed to be client-specific and each serve a mix of traffic including resource-consuming video conferencing, but also file sharing, cloud, messaging, etc.

The workload parameters for the scenarios in Table 5 are chosen to yield the following distributions of traffic shares among slices, assuming the user data rates in slices $\mathbf{R} = (1, 20, 10, 25, 50, 25, 25, 25)$. In scenario *SceMo6*, which roughly corresponds to the morning hours, the average distribution of traffic shares among slices in percentages is $(5, 40, 20, 5, 10, 20, 0, 0)$ with the average resource utilization of 56 %. Scenario *SceMo8* is different from scenario *SceMo6* only in that the workload of the corporate slice is distributed among three corporate slices 6, 7, and 8 in ratios 1/2, 1/4, and 1/4. Scenario *SceAN6* represents

the afternoon and yields the average traffic percentages of $(3, 30, 30, 12, 20, 5, 0, 0)$ with the utilization of 80 %. Finally, scenario *SceAN8* is similar to scenario *SceAN6*, but the workload of the corporate slice is equally distributed among three corporate slices 6, 7, and 8.

The contracted capacity shares, γ_s, for slices 1–6 are set approximately corresponding to the 99-percentiles of their traffic shares under $\nu = 3$, the maximum q_s over all the scenarios under consideration, and the user data rates $R_s^{\min}$. The contracted shares of all corporate slices are set γ_6 for simplicity. Thus, in this work, we assume that only about a half of the total capacity is contracted and the remainder is shared among slices without prioritization. Other scenarios can be designed based upon different assumptions as to the InP's business model and pricing strategies.

7 Numerical Results

In this section, we first numerically illustrate the main features of the considered slicing scheme. Then, an accuracy and performance assessment of the ML models under study is provided, preceded by a short discussion on the utilized data. Lastly, we address the impact of the cell capacity's variability on the efficiency of the slicing scheme.

7.1 Slicing Policy Features

We start with Fig. 5 providing insight into the considered slicing policy's features, namely slice isolation, data rate requirement enforcement and fairness. We consider capacity $\bar{C}$ and state $\mathbf{N} = (58, 22, 23, 2, 2, 9)$ corresponding to the average number of users in slices under scenario *SceMo6*, and vary the number of users in the video streaming slice. The voice slice receives a constant capacity share, i.e. $C_1 = \gamma_1 \bar{C}$, and the slicing schemes is applied to $\widetilde{\mathcal{S}} = \{2, \dots, 6\}$ and $\widetilde{C} = (1 - \gamma_1)\bar{C}$.

It can be observed in Fig. 5 (bottom) that as N_3 grows, capacity is fairly allocated to users accounting for their minimum and maximum data-rate requirements. For instance, for $N_3 \in [10, 40]$ the data rate of slice 3 is $R_3^{\max}$, while the rates in other $\widetilde{\mathcal{S}}$ slices, being greater than $R_3^{\max}$, are equal to each other. As soon as the data rates in slices $\widetilde{\mathcal{S}} \setminus \{3\}$ go below $R_3^{\max}$, all the rates decrease altogether, but not below their respective minima. However, as soon as N_3 grows beyond the contracted value, N_3^{cont}, the data rates in other slices do not change and only R_3 keeps decreasing so that the capacity of slice 3 remains equal to its value for

TABLE 5
Workload parameters of slices

			SceMo $\nu = 2.5$		SceAN $\nu = 2.7$	
Slice type	s	$\theta_{s[\text{sec}]}$	$q_{s[\%]}^{\text{Mo6}}$	$q_{s[\%]}^{\text{Mo8}}$	$q_{s[\%]}^{\text{AN6}}$	$q_{s[\%]}^{\text{AN8}}$
Voice	1	210	11.01	11.01	8.86	8.86
Best Effort	2	10	88.54	88.54	90.32	90.32
Streaming	3	3600	0.25	0.25	0.5	0.5
Gaming	4	1800	0.05	0.05	0.16	0.16
VR/AR	5	1800	0.05	0.05	0.13	0.13
Corporate	6	3600	0.1	0.05	0.03	0.01
Corporate	7	3600	0	0.025	0	0.01
Corporate	8	3600	0	0.025	0	0.01

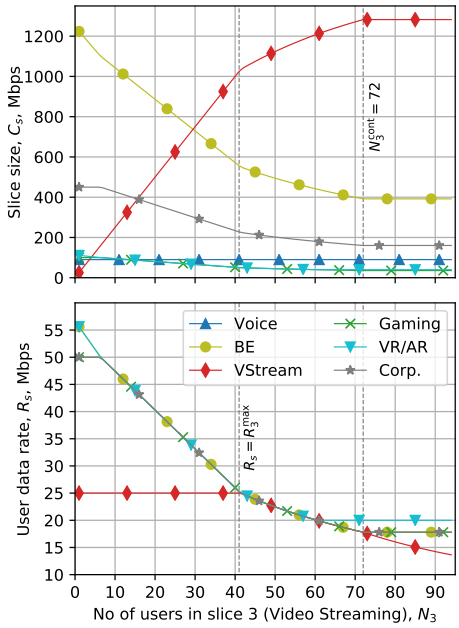


Fig. 5. Slice sizes (top) and user data rates (bottom) in states $\mathbf{N} = (58, 22, N_3, 2, 2, 9)$ vs. the number of users in slice 3.

N_3^{cont} (see Fig. 5 top). Thus, performance in slices $\widetilde{\mathcal{S}} \setminus \{3\}$ is isolated in the case of traffic increase in slice 3 beyond the contracted capacity.

7.2 ML Assessment Data and Tools

To numerically evaluate the accuracy and performance of the considered ML models, we rely on two types of labeled sample datasets. The majority of data is sampled via the slicing analysis simulator [44] with cell capacity values simulated via the procedure in Subsection 6.1. Here, separate datasets are obtained for the four scenarios using the parameter values in Tables 4 and 5. Another type of datasets – uniformly sampled – are populated with vectors $\mathbf{N}$ where each N_s is randomly sampled from $\{0, \dots, \lfloor \bar{C}/R_s^{\min} \rfloor\}$, and the average capacity is assumed. These are used to evaluate the offline learning setting.

Data were sampled and labelled with the voice slice omitted, i.e. assuming $\widetilde{\mathcal{S}} = \{2, \dots, S\}$ and $\widetilde{C} = (1 - \gamma_1)C$. All datasets have been filtered so as to consist of samples satisfying $\sum_{s \in \widetilde{\mathcal{S}}} N_s R_s^{\min} < \widetilde{C} < \sum_{s \in \widetilde{\mathcal{S}}} N_s R_s^{\max}$ only. Such states belong to set Ω^{opt} and require the time-greedy solution of the optimization problem $\text{SLICE}_{\mathbf{N},C}(\cdot)$.

To implement the considered ML models we utilized the corresponding functions from the *scikit-learn* library [45], namely *LinearRegression* with *PolynomialFeatures* preprocessing in the case polynomial regressions and *RandomForestRegressor* with 10 and 50 decision tree estimators. As for the ANN, we employed *Keras' Sequential* model with two ReLU-activated hidden layers of size 64 and 128. Such a configuration resulted from the hyperparameter optimization on a *SceAN6* dataset using the *KerasTuner* framework.

7.3 Accuracy of the ML Models

Accuracy assessment results are provided in Tables 6 and 7 and also in Fig. 6 and 7. Specifically, Table 6 shows the average accuracy for different evaluation settings. Row 1

TABLE 6
Accuracy of the ML models

		$S=6$					$S=8$				
		10-fold cross validation			Test on a SceMo dataset		10-fold cross validation			Test on a SceMo dataset	
	Technique	RMSE	MAE	MaxErr	MAE	AbsErrSD	RMSE	MAE	MaxErr	MAE	AbsErrSD
1. Trained on uniformly sampled data, $K=10^5$, averaged capacity $\bar{C}=2000$ Mbps	LinReg	50.66	33.71	864.80	142.43	136.46	30.73	20.55	605.38	95.92	100.39
	PolyReg2	45.07	28.53	860.93	72.19	64.90	27.29	17.12	631.14	99.63	82.35
	PolyReg4	28.15	17.51	1044.18	65.48	63.04	18.51	11.54	765.11	93.13	92.26
	RandFor10	16.88	8.54	946.22	69.33	62.14	18.94	11.36	479.04	99.17	87.48
	RandFor50	15.39	7.56	943.86	51.06	47.43	22.09	13.82	337.62	89.42	84.09
	ANN	11.34	7.20	616.10	76.22	76.43	7.43	4.77	238.60	115.45	104.16
2. Trained on a SceMo simulation dataset, $K=6\times10^4$	LinReg	25.98	19.17	366.83	18.83	28.38	20.01	14.65	451.77	14.69	24.34
	PolyReg2	21.89	16.34	394.66	16.01	26.01	15.95	11.70	402.74	11.71	22.88
	PolyReg4	21.12	15.69	433.93	15.46	26.02	15.45	11.29	520.43	11.28	22.88
	RandFor10	24.43	17.81	400.72	19.64	29.94	18.19	13.02	376.50	20.08	27.23
	RandFor50	23.83	17.38	395.62	18.90	29.29	17.41	12.46	380.63	18.66	25.91
	ANN	21.68	16.07	413.95	16.10	25.95	16.01	11.69	394.16	13.33	23.83
3. Trained on a SceAN simulation dataset, $K=6\times10^4$	LinReg	20.32	15.16	312.64	118.30	108.40	15.01	11.11	292.35	84.07	98.34
	PolyReg2	17.54	13.34	215.59	52.61	41.86	12.68	9.59	209.57	38.12	38.84
	PolyReg4	16.76	12.65	225.66	75.74	120.72	12.13	9.06	220.98	53.76	113.44
	RandFor10	19.29	14.36	258.00	118.28	100.20	14.00	10.32	254.45	89.97	72.43
	RandFor50	18.69	13.92	253.86	121.32	101.42	13.44	9.91	228.91	88.27	73.26
	ANN	17.69	13.54	270.44	77.79	47.15	13.02	9.85	212.46	57.84	43.94

deals with offline learning. Here, the models are trained on uniformly sampled data with averaged cell capacity and then tested on *SceMo* simulation datasets. Rows 2 and 3 illustrate the online learning setting. In row 2 the models are trained on *SceMo* datasets different from the test ones, whereas in row 3 the models are trained on *SceAN* datasets and then tested on the test *SceMo* datasets. The same *SceMo6* and *SceMo8* datasets are used for testing in all rows.

It can be observed that the online learning setting yields substantially better results than the offline one as long as training and testing are performed on data sampled with the same workload parameter values. However, when tested on data sampled with differing workload parameters, the accuracy is generally quite poor, although some models generalize better than others.

We specifically look into the sensitivity of the models' accuracy to the change in the workloads in Fig. 6 and 7. As one may observe, the accuracy of the models is hardly impaired by variations in the overall workload level (Fig. 6), but is greatly affected by changes in the workload distribution among slices (Fig. 7). We clearly see that the polynomial regressions closely followed by ANN demonstrate the best generalization capacity among the considered models.

A comparison between $S=6$ and $S=8$ results provided in Table 6 shows that an increase in the number of slices improves the prediction accuracy in the online training setting. This result is particularly welcome since it is for a larger number of slices that the exact solution algorithm becomes slow and an alternative solution is needed.

Whereas the metrics in Table 6 are averaged over all slices, Table 7 provides accuracy assessment by slice. It can be observed that all the models simultaneously exhibit substantial variations in prediction accuracy from slice to slice. Unfortunately, these variations cannot be explained and hence predicted from the slices' SLA parameters: e.g., the video streaming slice yields the best accuracy on *SceMo* data and the worst on *SceAN*. On the other hand, we notice that for each workload range, significant errors come from about two slices, while approximations for others are satisfactory. Thus, in the ML pipeline it may suffice to monitor and adjust predictions for just a few slices.

7.4 Performance of the ML Algorithms

Besides accuracy, we evaluate performance of the considered models by looking into their prediction time (Table 8), trade-off between accuracy and prediction time (Fig. 8), training dataset sizes (Fig. 9) and capability to respect the data rate constraints (Table 9).

According to Table 8, the regression models are substantially faster than the random forests and ANN. However, when increasing the number of slices from $\widetilde{S}=5$ to $\widetilde{S}=7$, the execution time of the regressions approximately doubles, while that of RF and ANN barely changes. It should be noted that the prediction time of regressions and ANN is determined by the model dimensions, whereas that of RF also depends on the data and varies from one dataset to another.

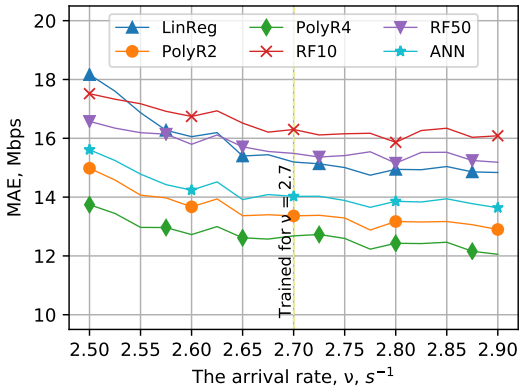


Fig. 6. Accuracy sensitivity to the overall workload level. For each run, the models are trained on a *SceAN6* sample ($\nu=2.7$) of size 6×10^4 and then tested on samples of size 250 generated for $\nu\in[2.5,2.9]$ under the *SceAN* scenario assumptions. Lines show MAE averaged over 10 runs vs. ν.

TABLE 7
Accuracy of the ML models by slice via 10-fold cross validation

Dataset	Technique	Best Effort RMSE	Best Effort MAE	Streaming RMSE	Streaming MAE	Gaming RMSE	Gaming MAE	VR/AR RMSE	VR/AR MAE	Corporate RMSE	Corporate MAE
SceMo6	LinReg	68.49	53.18	4.76	1.31	11.76	8.42	14.11	9.47	30.77	23.46
	PolyReg2	64.24	50.18	4.36	1.57	7.25	4.92	8.54	5.77	25.08	19.24
	PolyReg4	63.87	49.90	3.38	0.94	6.85	4.57	7.73	5.10	23.77	17.92
	RandFor10	73.40	57.16	4.03	0.50	8.02	5.12	9.24	5.91	27.48	20.34
	RandFor50	71.69	55.84	3.96	0.50	7.78	4.99	8.95	5.75	26.76	19.84
	ANN	64.87	50.81	4.80	1.32	7.02	4.79	8.07	5.42	24.06	17.99
SceAN6	LinReg	29.27	21.71	44.91	35.41	11.86	8.27	9.66	6.50	5.93	3.92
	PolyReg2	26.24	19.94	41.84	33.18	9.06	6.55	6.67	4.44	3.87	2.58
	PolyReg4	25.37	19.25	40.40	31.59	8.49	6.18	6.00	3.89	3.55	2.34
	RandFor10	29.17	22.05	46.11	35.51	9.90	7.11	7.08	4.42	4.24	2.71
	RandFor50	28.23	21.34	44.70	34.45	9.58	6.89	6.86	4.30	4.08	2.61
	ANN	26.29	20.00	42.21	33.24	8.84	6.39	6.31	4.32	3.79	2.60

Fig. 8 reveals the trade-off between accuracy and prediction time based on the data in Tables 6 and 8 for the online learning scenario. Here, the XGBoost regression model [46] is added as a benchmark. It can be observed that PolyReg2 provides the best compromise between execution time and accuracy.

Fig. 9 shows the learning curves in terms of MAE for the models under study. We observe that the polynomial regressions perform the best in the online learning setting, with PolyReg2 potentially slightly less accurate than PolyReg4, but reaching its maximum accuracy faster, with less than 1500 samples. The accuracy of the random forest regressors and ANN keeps improving for much larger training dataset sizes, especially when learning offline. However, as we can see from Table 6, good cross-validation results on a uniformly sampled dataset with a fixed capacity value, exhibited by ANN and the RF regressors, do not guarantee satisfactory prediction accuracy on a test dataset simulated with varying cell capacity and hence noisy.

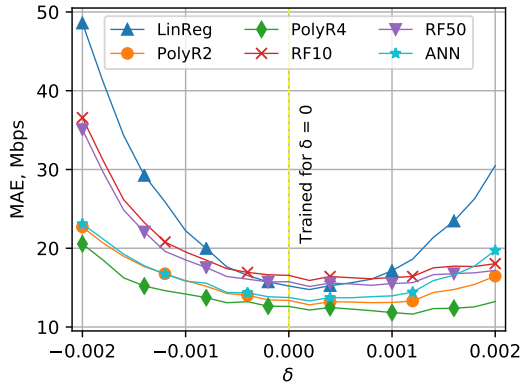


Fig. 7. Accuracy sensitivity to workload variations among slices. For each run, the models are trained on a *SceAN6* sample of size 6×10^4 and then tested on samples of size 250 generated for $q_2 = q_2^{\mathrm{AN5}} - \delta$, $q_3 = q_3^{\mathrm{AN5}} + \delta$, $\delta \in [-0.2\%, 0.2\%]$. Lines show MAE averaged over 10 runs vs. δ.

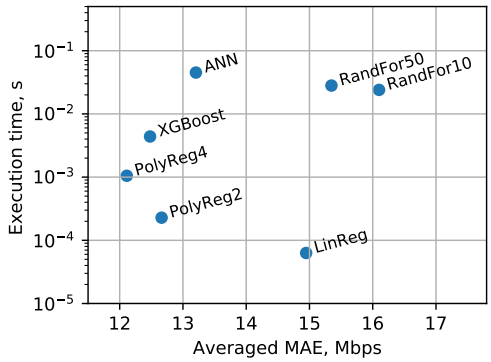


Fig. 8. Trade-off between accuracy and prediction time for the online learning scenario based on data in Tables 6 and 8.

TABLE 8
Prediction time per 1000 of data points, s

	$S = 6$ ($\widetilde{S} = 5$)	$S = 8$ ($\widetilde{S} = 7$)
LinReg	6.30×10^{-5}	1.11×10^{-4}
PolyReg2	2.28×10^{-4}	5.03×10^{-4}
PolyReg4	1.05×10^{-3}	2.01×10^{-3}
RandFor10	2.40×10^{-2}	2.80×10^{-2}
RandFor50	2.81×10^{-2}	2.94×10^{-2}
ANN	4.51×10^{-2}	4.47×10^{-2}
Exact solution	4.35	2.06×10^{2}

Finally, Table 9 shows that the approximations via all the considered ML models are quite prone to SLA violation, especially under higher workloads. Therefore, whatever the adopted model, a specific SLA guarantee control must be implemented in the ML pipeline by means of a policy node. Such a node might first check whether the inequality $C_s \frac{C}{\sum_{m\in\mathcal{S}} C_m} \geq C_s^{\min}(N_s)$, where C_s is the size of slice s returned by the model node, holds for all $s \in \mathcal{S}$, and if this is not the case then set $C_s = \gamma_s C$ or $C_s = C_s^{\min}(N_s)\frac{C}{\sum_{m\in\mathcal{S}} C_m^{\min}(N_m)}$. Either of these would provide a suboptimal yet SLA-conform allocation. In our modeling setup adding such operations would increase the execution time by about 1.32×10^{-2} s for $S = 6$ and 1.4×10^{-2} s for $S = 8$ per 1000 data points.

7.5 Impact of Varying Traffic and Channel Conditions

The considered slicing scheme yields the optimal resource allocation to slices provided that it is executed to the current cell capacity value. A delay in re-slicing may result in issues

TABLE 9
Probability that the approximation leads to SLA violation, %

	LinReg	PReg2	PReg4	RF10	RF50	ANN
SceMo	10.30	12.79	10.90	00.50	00.29	11.65
SceAN	19.29	23.31	17.38	19.68	18.14	15.08

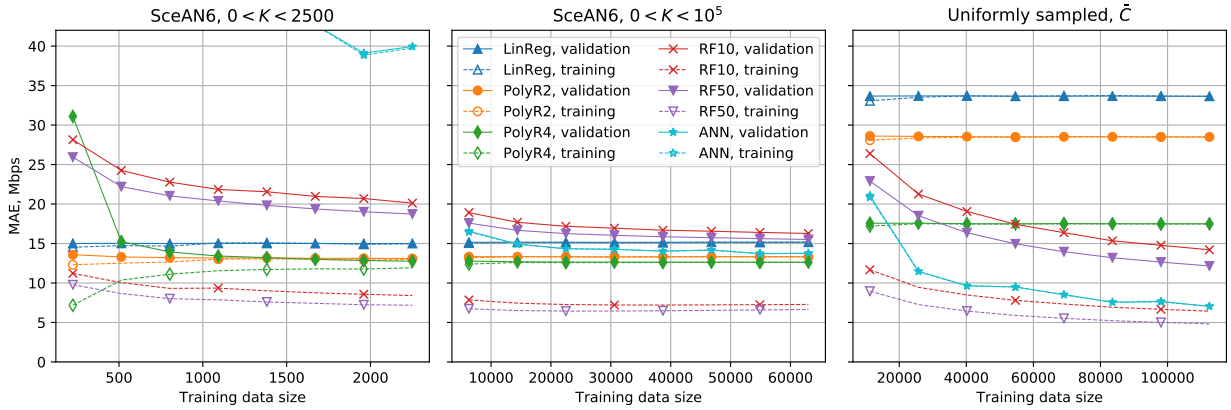


Fig. 9. Learning curves via 10-fold cross validation over data sampled from *SceAN* (left and center) and uniformly with the averaged capacity (right), $S = 6$. Solid lines show MAE from validation data, dashed lines represent MAE from training data.

illustrated in Fig. 10. Here user data rates for $\mathbf{N}$ corresponding to the average numbers of users in *SceMo* (top) and *SceAN* (bottom) are plotted vs. C. Solid lines show the data rates computed for each plotted value of C and dashed lines indicate the data rates computed for $\bar{C}$ and then scaled for each C with factor $C/\bar{C}$. It can be seen in Fig. 10 (top) that the scaled data rate in the video streaming slice for $C > \bar{C}$ is above the maximum, $R_3^{\max} = 25$, which results in a waste of resources. A bigger issue can be observed in Fig. 10 (bottom). Here, the scaled data rate in the VR/AR slice goes below the required $R_5^{\min} = 20$, which leads to SLA violation.

To investigate further the impact of capacity variations, in Fig. 11 we plot vs. the arrival rate the probabilities that the minimum data rates, although maintained under $\bar{C}$, are violated under a current capacity value provided that re-slicing is done in real time. Note that here we do not account for the contracted numbers of users, and this can hardly result in SLA violation, for which we need $\mathbf{N}^{\min}\mathbf{R}^{\min} > \bar{C} + \xi$ rather than $\mathbf{N}\mathbf{R}^{\min} > \bar{C} + \xi$. We observe that under the *SceAN* user distribution this probability becomes significant when the workload initially assumed in the scenario is multiplied by 3.7, and under *SceMo* the workload has to be multiplied by 7. Both probabilities attain about 6–7 % and then go down as the workload grows further because $\bar{C}$ also becomes insufficient for the growing traffic.

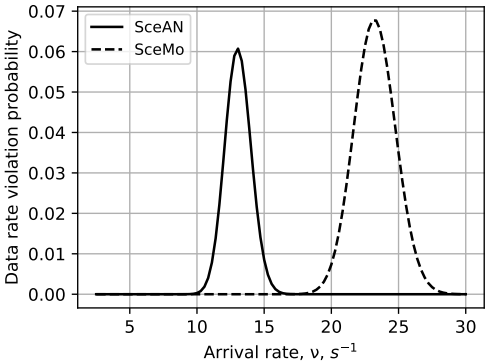


Fig. 11. Probabilities of the minimum data rates' violation due to capacity variations vs. ν. The lines represent $\mathrm{P}\{\mathbf{N}, \xi : \bar{C} + \xi < \mathbf{N}\mathbf{R}^{\min} \le \bar{C}\}$ for user distributions $\mathbf{q}^{\mathrm{Mo6}}$ (dashed) and $\mathbf{q}^{\mathrm{AN6}}$ (solid).

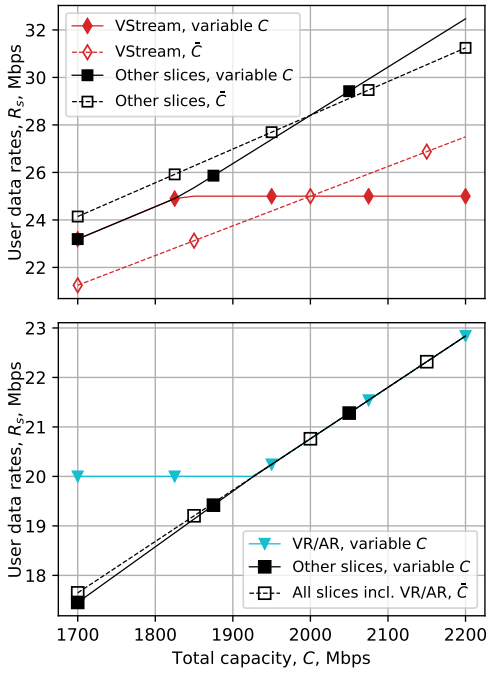


Fig. 10. User data rates vs. cell capacity in the system states corresponding to the average numbers of users in the *SceMo* (top) and *SceAN* (bottom) scenarios. Solid lines represent the data rates computed by Algorithm 1 for each plotted capacity value C; dashed lines show the data rates computed by Algorithm 1 for $\bar{C} = 2000$ and then, for each C, scaled with factor $C/\bar{C}$.

The data provided in Fig. 10 and Fig. 11 emphasize that the variable cell capacity may lead to significant degradation of contracted rates when re-slicing is not performed timely. Otherwise, its impact is limited to very high workload regimes and barely affects SLA. Complementing these illustrations is Fig. 12 showing the system degradation probability for different re-slicing triggers. By the degradation probability we understand the probability that the data rate requirements of at least one slice are violated. Separate simulations are performed for (i) the variable cell capacity sampled at 1 s intervals, (ii) the approximation via the constant cell capacity $\bar{C} = 2000$, and (iii) the variable cell capacity for degradation control and $\bar{C}$ for re-slicing. By analyzing the presented data, one observes that the least degradation is induced by invoking re-allocations upon both session arrival and departures. Re-slicing upon arrivals only shows a performance comparable to that of

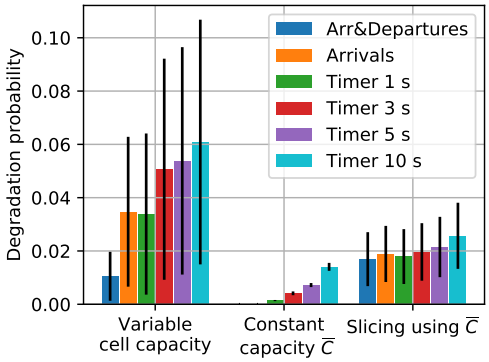


Fig. 12. Estimated probabilities of the minimum data rates' violation for different re-slicing triggers under the *SceAN6* scenario. Colored bars represent $\frac{1}{T}\int_0^T 1\{\exists s \in \mathcal{S} : C_s(t) < N_s(t)R_s^{\min}\}dt$, where $N_s(t)$ is the number of sessions in slice s at time t and $C_s(t) = C_s(t^*)\frac{C(t)}{C(t^*)}$ with t^* being the time of the last re-slicing before t. The values are estimated by simulation for $T \approx 8$ hours and averaged over 10 runs. Black bars indicate STD over 10 runs.

the timer-based approach with 1 s intervals in the depicted *SceAN* scenario characterized by higher workloads. Quite naturally, an increase in the re-slicing interval results in a higher performance degradation. Importantly, we observe that relying upon a constant cell capacity leads to overly optimistic results in high load conditions. At the same time, for lower load conditions corresponding to the *SceMo* scenario (not shown in the figure) no significant difference can be seen between the three considered parameterizations. Thus, one can conclude that capturing the cell capacity dynamics timely is critical in high load conditions.

8 Conclusions

Motivated by critical time and accuracy constraints for the slicing process in future 5G NR systems, in this paper we have compared and evaluated performance of ML algorithms for enhancing a RAN slicing scheme as standardized by ITU-T. By accounting for a realistic channel model and slice workload distributions, we have assessed their accuracy and efficiency as well as sensitiveness to cell rate variations operating in online and offline learning regimes.

We have shown that the online implementation exhibits improvements over the offline for the same slice composition, overall workload level and workload distribution among slices. Furthermore, the polynomial regressions are potentially our best choice for online learning, since in this setting they outperform both neural and random forest algorithms in terms of accuracy, execution time, sensitiveness to workload variations and the size of training data needed to achieve the optimal accuracy, which makes them suitable to accommodate frequently changing traffic distributions across slices. However, the random forest regressor is a close competitor capable of achieving a better accuracy when trained offline, although with a much larger training dataset.

By assessing the effect of the overall workload level and channel variations we have demonstrated that the latter may lead to significant degradation of the contracted data rates, especially in high workload regime, if re-slicing is not performed timely, which drives the need for ML enhancement. Although when using an ML technique a monitoring and adjustment mechanism is needed in the ITU-T standardized ML pipeline (i.e., the policy node) to enforce SLA constraints.

We note that the proposed approach can be adapted to other RAN slicing schemes formulated as optimization problems with constraints. The presented framework, results and discussion could hence provide guidance for such an adaptation.

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, 2017.

[2] V. Petrov, M. A. Lema, M. Gapeyenko, K. Antonakoglou, D. Moltchanov, F. Sardis, A. Samuylov, S. Andreev, Y. Koucheryavy, and M. Dohler, "Achieving end-to-end reliability of mission-critical traffic in softwarized 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 485–501, 2018.

[3] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, 2017.

[4] 3GPP, "5G; System architecture for the 5G system," ETSI, 3GPP TS 23.501 version 15.2.0 Release 15, 2018. [Online]. Available: http://www.etsi.org/deliver/etsits/123500123599/123501/15.02.0060/ts123501v150200p.pdf

[5] N. Yarkina, L. M. Correia, D. Moltchanov, Y. Gaidamaka, and K. Samouylov, "Multi-tenant resource sharing with equitable-priority-based performance isolation of slices for 5G cellular systems," *Comp. Commun.*, vol. 188, pp. 39–51, 2022.

[6] Y. Koucheryavy, E. Lisovskaya, D. Moltchanov, R. Kovalchukov, and A. Samuylov, "Quantifying the millimeter wave new radio base stations density for network slicing with prescribed SLAs," *Computer Communications*, vol. 174, pp. 13–27, 2021.

[7] H. H. Mahmoud and T. Ismail, "A review of machine learning use cases in telecommunication industry in the 5G era," in *16th ICENCO*, 2020.

[8] M. Zorzi, A. Zanella, A. Testolin, M. De Filippo De Grazia, and M. Zorzi, "Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence," *IEEE Access*, vol. 3, pp. 1512–1530, 2015.

[9] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE INFOCOM*, 2019.

[10] P. Sun, N. Aljeri, and A. Boukerche, "Machine learning-based models for real-time traffic flow prediction in vehicular networks," *IEEE Netw.*, vol. 34, no. 3, pp. 178–185, 2020.

[11] ITU-T Rec. Y.3172, "Architectural framework for machine learning in future networks including IMT-2020," 2020.

[12] ITU-T Rec. Y.3174, "Framework for data handling to enable machine learning in future networks including IMT-2020," 2020.

[13] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Elsevier Computer Networks*, vol. 167, 2020.

[14] A. Banchs, G. de Veciana, V. Sciancalepore, and X. Costa-Perez, "Resource allocation for network slicing in mobile networks," *IEEE Access*, vol. 8, pp. 214 696–214 706, 2020.

[15] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration on automation of 5G network slicing with machine learning," in *Machine Learning for a 5G Future (ITU K)*, 2018, pp. 1–8.

[16] M. H. Abidi, H. Alkhalefah, K. Moiduddin, M. Alazab, M. K. Mohammed, W. Ameen, and T. R. Gadekallu, "Optimal 5G network slicing using machine learning and deep learning concepts," *Computer Standards & Interfaces*, vol. 76, p. 103518, 2021.

[17] N. Kumar and A. Ahmad, "Machine learning-based QoS and traffic-aware prediction-assisted dynamic network slicing," *International Journal of Commun. Netw. and Distr. Syst.*, vol. 28, no. 1, pp. 27–42, 2022.

[18] N. Salhab, R. Langar, R. Rahim, S. Cherrier, and A. Outtagarts, "Autonomous network slicing prototype using machine-learning-based forecasting for radio resources," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 73–79, 2021.
[19] M. Toscano, F. Grunwald, M. Richart, J. Baliosian, E. Grampín, and A. Castro, "Machine learning aided network slicing," in *2019 21st International Conference on Transparent Optical Networks (ICTON)*. IEEE, 2019, pp. 1–4.
[20] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.
[21] M. Yan, G. Feng, J. H. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent resource scheduling for 5G radio access network slicing," *IEEE Trans. on Veh. Technol.*, vol. 68, no. 8, pp. 7691–7703, 2019.
[22] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah, and W. Jiang, "Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks," *IEEE Access*, vol. 7, pp. 45 758–45 772, 2019.
[23] H. Wang, Y. Wu, G. Min, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach," *Information Sciences*, vol. 498, pp. 106–116, 2019.
[24] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz (Release 14)," 3GPP, 3GPP TR 38.901 V14.1.1, July 2017.
[25] ——, "Derivation of test tolerances and measurement uncertainty for User Equipment (UE) conformance test cases," 3GPP, 3GPP TR 38.903 V16.4.0, December 2018.
[26] T. Bai, A. Alkhateeb, and R. W. Heath, "Coverage and capacity of millimeter-wave cellular networks," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 70–77, 2014.
[27] V. Petrov, M. Komarov, D. Moltchanov, J. M. Jornet, and Y. Koucheryavy, "Interference and SINR in millimeter wave and terahertz communication systems with blocking and directional antennas," *IEEE Trans. on Wir. Comm.*, vol. 16, no. 3, pp. 1791–1808, 2017.
[28] R. Kovalchukov, D. Moltchanov, A. Samuylov, A. Ometov, S. Andreev, Y. Koucheryavy, and K. Samouylov, "Evaluating SIR in 3D millimeter-wave deployments: Direct modeling and feasible approximations," *IEEE Trans. on Wireless Commun.*, vol. 18, no. 2, pp. 879–896, 2019.
[29] G. R. MacCartney, T. S. Rappaport, and S. Rangan, "Rapid fading due to human blockage in pedestrian crowds at 5G millimeter-wave frequencies," in *GLOBECOM 2017*. IEEE, 2017, pp. 1–7.
[30] M. Gapeyenko, A. Samuylov, M. Gerasimenko, D. Moltchanov, S. Singh, E. Aryafar, S.-p. Yeh, N. Himayat, S. Andreev, and Y. Koucheryavy, "Analysis of human-body blockage in urban millimeter-wave cellular communications," in *IEEE ICC*, 2016, pp. 1–7.
[31] V. Petrov, M. Komarov, D. Moltchanov, J. M. Jornet, and Y. Koucheryavy, "Interference and SINR in millimeter wave and terahertz communication systems with blocking and directional antennas," *IEEE Trans. on Wireless Commun.*, vol. 16, no. 3, pp. 1791–1808, 2017.
[32] C. Balanis, *Antenna theory: analysis and design*. Wiley, 2016, 4th ed.
[33] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," in *IEEE 24th INFOCOM*, vol. 3, March 2005, pp. 1897–1907.
[34] 3GPP, "NR; Physical channels and modulation (Release 15)," 3GPP TR 38.211, Dec 2017.
[35] GSM Association Official Document NG.116 (11/2020), "Generic network slice template," version 4.0, 2020. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v4.0-1.pdf
[36] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, and B. Bellalta, "Usage of network simulators in machine-learning-assisted 5G/6G networks," *IEEE Wireless Commun.*, vol. 28, no. 1, pp. 160–166, 2021.
[37] F. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecomm.*, vol. 8, no. 1, pp. 33–37, 1997.
[38] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1992.
[39] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.
[40] Scikit-learn Developers. Decision trees – scikit-learn 1.0.2 user guide. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html
[41] Sandvine, "The mobile internet phenomena report," Waterloo, Canada, Tech. Rep., May 2021. [Online]. Available: https://www.sandvine.com/download-mobile-internet-phenomena-report-2021
[42] Ericsson, "5G for gaming." Accessed 14 Apr. 2022. [Online]. Available: https://www.ericsson.com/en/5g/5g-for-gaming
[43] 3GPP, "Virtual reality (VR) media services over 3GPP," 3GPP TR 26.918 V16.0.0, December 2018.
[44] N. Polyakov, N. Yarkina, and K. Samouylov, "A simulator for analyzing a network slicing policy with SLA-based performance isolation of slices," *Discrete and Continuous Models and Applied Computational Science*, vol. 29, no. 1, pp. 36–52, 2021.
[45] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[46] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.

Natalia Yarkina received her M.Sc. (Hons) in applied mathematics and Cand.Sc. in physics and mathematics from RUDN University, Moscow, Russia, in 2004 and 2007 respectively, and the M.Sc. (Dist) degree in translation studies from the University of Aberdeen, UK, in 2016. She is now pursuing her Ph.D. degree in computing and electrical engineering at TAU. Her present research focus is on stochastic modeling, performance analysis and efficient simulation of wireless communication networks. She coauthored four books on business process management in telecommunications and teletraffic theory, and over 20 research articles and conference papers on telecommunication network performance evaluation.

Anna Gaydamaka received the B.Sc. (Hons) degree in Business Informatics from the Peoples' Friendship University of Russia (RUDN University) in 2018 and the M.Sc. degree in Computer Science, program Data Science and Business Informatics) from Universita di Pisa, Italy in 2021. Currently, she is pursuing a Ph.D. in Computing and Electrical Engineering and working as a Doctoral Researcher at Tampere University, Finland. Her research interests include resource planning of the fifth and sixth-generation wireless network, mathematical models for performance KPIs optimization and machine learning techniques.

Dmitri Moltchanov is a university lecturer at TAU. He received the M.Sc. (2000) and Cand.Sc. (2003) degrees from the St. Petersburg State University of Telecommunications, Russia, and the Ph.D. (2006) degree from TUT. He has authored more than 150 publications and has taught more than 60 courses on several topics in telecommunications. His current research interests include research and development of 5G/5G+ systems, industrial IoT applications, mission-critical V2V/V2X systems, and blockchain technologies.

Yevgeni Koucheryavy received the Ph.D. degree from the Tampere University of Technology (TUT), Finland. He is currently a Professor at the Laboratory of Electronics and Communications Engineering, TUT. He is the author of numerous publications in the field of advanced wired and wireless networking and communications. His current research interests include various aspects in heterogeneous wireless communication networks and systems, the Internet of Things and its standardization, and nanocommunications. He is Associate Technical Editor of the IEEE Communications Magazine and an Editor of the IEEE Communications Surveys and Tutorials.