Comparison of Machine Learning Algorithms for Priority-Based Network Slicing in 5G Systems

Anna Gaydamaka
Higher School of Economics National Research University
Moscow, Russia
agajdamaka@hse.ru

Natalia Yarkina
Unit of Electrical Engineering Tampere University
Tampere, Finland
natalia.yarkina@tuni.fi

Viktoriia Khalina
Higher School of Economics National Research University
Moscow, Russia
vhalina@hse.ru

Dmitri Moltchanov
Unit of Electrical Engineering Tampere University
Tampere, Finland
dmitri.moltchanov@tuni.fi

***Abstract*—Network slicing is a technique to enable multi-tenant operation in future 5G systems. Efficient implementation of slicing at the air interface requires comprehensive optimization algorithms characterized by high execution complexity. To address this issue in the paper, we first present a priority-based mechanism enabling performance isolation between slices competing for resources. Then, to speed up the resource arbitration process under high traffic conditions, when resource shares need to be re-calculated in sub-second timescales, we propose and compare several machine learning techniques: linear regression, polynomial regression, a random forest regressor, and a two-layer artificial neural network. The techniques' performance is assessed by utilizing the mean squared error. Our results show that a high order polynomial regression provides the desired balance between computational complexity and accuracy, outperforming both the simpler linear regression and the more complex random forest and neural network algorithms.**

I. Introduction

The introduction of 5G cellular architecture not only increases resource use efficiency at the air interface, but will also enable flexibility of end-to-end resource control and management [1], [2]. One of these advanced functionalities is network slicing providing the tools for end-to-end resource management, including the wireless access interface [3].

Following [4], a network slice is defined as a logical network that provides specific network capabilities and network characteristics. The document also demands slice isolation, although defined in a broad sense encompassing multiple levels, e.g., 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 air interface, where channel conditions need to be accounted for when designing efficient isolation schemes.

Up to date, a number of algorithms have been proposed for network slicing at the air interface with various performance isolation criteria taken into account, see Section II for a review. As most of those approaches formalize and solve an optimization problem, the solution complexity becomes a critical issue. In dynamically changing wireless channel conditions, ensuring both isolation and fairness of resource allocation may lead to the inability to timely redistribute resources between slices and flows that belong to different slices. As a result, lightweight approximations of exact solutions could prove crucial for practical implementations.

In this paper, we first formalize a model of resource slicing at the air interface aimed at fair priority-based isolation of slices [5]. Then, to reduce the solution complexity, we propose machine learning (ML) techniques for the efficient solution of the optimization problem. Particularly, we consider and analyze the performance of four candidate algorithms: linear regression, polynomial regression, the random forest regressor, and the two-layer artificial neural network (ANN). The main conclusions of our work can be formulated as follows:

- supervised learning techniques are able to provide suitable approximations for the resource allocation process, ensuring both slice performance isolation and fairness;
- a comparison of ML approximations shows that the balance between accuracy and complexity is provided by a high-order polynomial regression.

The rest of the paper is organized as follows. In Section II we review the recent work related to network slicing and two approaches to it: reservation techniques and priorities. Section III presents the system model and its components. In Section IV we provide the exact solution algorithms and discuss ML techniques for speeding up resource arbitration. The numerical results and their interpretation are provided in Section V. Conclusions are drawn in the last section.

II. Background and Related Work

Overall, the algorithms and solutions for performance isolation of slices at the air interface can be divided into three categories: (i) priority-based solutions, (ii) reservation-based solutions, and (iii) a mixture of these. The study in [7] provides a survey of resource allocation for network slicing.

The logarithmic utility maximization of a two-tier heterogeneous wireless network for various vehicular applications is considered in [8]. The authors present a joint optimization solution for bandwidth reservation and priority-based slice allocation. Computer simulations show that the proposed priority-based reservation and slice allocation scheme uses network resources more efficiently than traditional ones. The authors of [9] look for an optimal scheduling policy for a wireless network. In cases of high computational complexity tasks where traditional optimization techniques cannot be

applied, they propose to use a priority-based, "most energy-efficient resource first" policy. Simulation results show that the proposed policy is scalable, reliable, and achieves significant improvement in energy efficiency.

The paper [10] introduces a scheme for dynamic resource allocation based on priorities. Based on the priorities and demand profiles of slices, the agent dynamically allocates resources between slices. The optimization problem is solved using linear programming. The proposed policy proves efficient in terms of QoS and resource use efficiency. In [11] the authors propose a heuristic method based on an admission control mechanism that dynamically allocates network resources between slices. Simulations show that the proposed method uses network resources more efficiently and gives high scalability when the number of users in each slice increases. In [12] the authors focus on a challenging problem of resource isolation in multi-hop wireless networks similar to integrated access and backhaul architecture [13]. The authors propose to utilize the shortest paths for higher priority traffic. In lower-priority slices, transport flows are routed to minimize the weighted sum of interference for other slices. The proposed slice control scheme significantly increases the average slice throughput and reduces the average slice delay.

To reduce the computational complexity, several ML-based approaches have been proposed for network slicing at the air interface. In their study [14], the authors investigate whether it is worth applying deep reinforcement learning in the network resource management schemes. They consider two network slicing scenarios: radio resource slicing and priority-based core network slicing. The simulations show that deep learning enhances the effectiveness and agility of network slicing. The authors in [15] propose a system for dynamic reservation of unused resources for next-generation mobile networks based on deep reinforcement learning algorithms. They show that by tuning the objective function, significant improvements in resource utilization are achieved. However, no guidance is provided on the choice of these functions.

III. System Model

In this section, we introduce our system model and its components. We start 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 scheme. Finally, we define the metrics of interest.

1) Radio Access and Resources: Consider a base station (BS) providing the virtualization of radio access resources and network slicing. We study the downlink transmission of such a BS. The considered network structure corresponds to a heterogeneous network of a single operator – the infrastructure provider (InP). We consider a network of one InP.

The BS has a number of radio access technologies (RAT), denoted by RAT A, RAT B, RAT C, etc. Each RAT X, $X = A, B, C, ...$, has resource blocks allocated to users, see Fig. 1. The capacity of resource blocks is denoted by $\mathbf{Q}_X = (Q_{X_{m[Gbps]}})_{m=1,...,M_X}$, where M_X represents the number of different channels in RAT X. The corresponding Radio

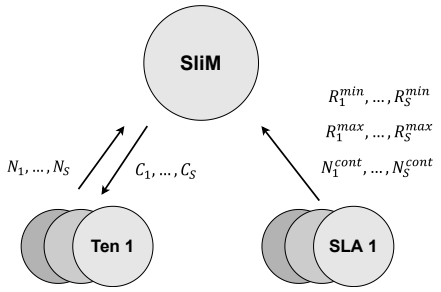


Fig. 1. Illustration of the considered slicing model.

Resource Management (RRM) entity controls the resources of each RAT. The maximum capacity of RRM is denoted by $K_{X[Gbps]}$, $X = A, B, C, ...$ Further, a Common Resource Manager (CoRM) globally controls the RRMs. The slicing Manager (SliM) is assumed to be responsible for aggregating the capacities of each RAT to an overall capacity of the BS, denoted by $C_{[Gbps]}$. Another function of SliM is to distribute C among slice tenants (Ten 1, Ten 2, ..., Ten S).

Each RAT has an assigned frequency band. Its maximum capacity is achieved for the best Adaptive Modulation and Coding (AMC) conditions and Multiple-Input-Multiple-Output (MIMO) order, among other factors. The actual data rate available to users depends on the radio channel propagation conditions, as the data rate of each physical channel varies over time depending on the Signal to Interference plus Noise Ratio (SINR). Using these parameters, the session rate can be related to the amount of requested resources. In what follows, we consider the total maximum capacity, $C_{[Gbps]}$, when optimizing the resource allocation. This parameter can be adjusted to the network dynamics if needed. It is assumed that the resulting slicing distribution is valid over a time interval in which network conditions do not change drastically.

2) Traffic and Slices: We assume that there are S slices at the BS. The set of all slices is denoted by $\mathcal{S}$, $|\mathcal{S}| = S$. Let $C_s \geq 0$ denote the capacity of slice $s \in \mathcal{S}$. We assume that each slice is responsible for one type of service (e.g., video streaming, video conferencing, gaming). Thus, the traffic in each slice is homogeneous. Considering that C is the total capacity of the BS, the capacities of slices are such that $\sum_{s\in\mathcal{S}} C_s \leq C$.

Let N_s denote the number of users in slice s and let the data rate R_s given to each user in slice s be the result of the even distribution of the slice's resource, i.e.,

$$R_s = C_s/N_s, \ s \in \mathcal{S}. \tag{1}$$

The row vector containing the numbers of users in all slices is denoted by $\mathbf{N} = (N_s)_{s\in\mathcal{S}}$. We underline that it is assumed that each user has only one connection in only one slice. If a user has multiple connections, he or she is considered and served as multiple users, one per connection. The column vector of data rates is denoted by $\mathbf{R} = (R_s)_{s\in\mathcal{S}}$.

We further propose two predefined thresholds to satisfy the SLA requirements between the InP and the slice tenant: R_s^{min} and R_s^{max}. As long as the number of users in the slice does not exceed a contracted number N_s^{cont}, i.e. for $N_s \leq N_s^{cont}$, the data rate must not go below the predefined threshold, i.e.,

$0 < R_s^{min} \leq R_s$. The R_s^{min} threshold corresponds to the minimum required data rate to meet the QoS requirements of the service provided in slice s. It is assumed that a user cannot get a proper service if the data rate is below this value. In any case, we assume that R_s^{min} is agreed upon between the InP and the slice tenant. We also introduce a row vector $\mathbf{N}^{cont} = (N_s^{cont})_{s\in\mathcal{S}}$.

The second threshold, R_s^{max}, is the maximum user data rate for a slice, $R_s \leq R_s^{max} \leq C$, determined by the service provided in the slice. It corresponds to such a value that allocating a data rate higher than this will not result in any gain in QoS for the user. The value of R_s^{max} is agreed upon between the InP and the slice tenant. Let $\mathbf{R}^{min} = (R_s^{min})_{s\in\mathcal{S}}$, $\mathbf{R}^{max} = (R_s^{max})_{s\in\mathcal{S}}$ be column vectors.

3) Dynamic Priority-Based Slice Isolation: 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 that slice does not exceed the contracted threshold $0 \leq N_s^{cont} \leq \lfloor C/R_s^{min} \rfloor$. Thus, the InP guarantees the performance isolation of slice s by providing it with at least

$$C_s^{guar} = \min\{N_s, N_s^{cont}\} R_s^{min}. \tag{2}$$

Any capacity remaining after allocating C_s^{guar} to each slice 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, $\mathbf{N}^{cont}\mathbf{R}^{min}$, 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 $\mathbf{N}^{cont}$, and (ii) their demand expressed in terms of the number of users $\mathbf{N}$, see Fig.1. Flexibility is assured by the fact that when some slices do not use all their contracted capacity $N_s^{cont}R_s^{min}$, the remaining capacity $(N_s^{cont} - N_s)R_s^{min}$ becomes available to other slices if they need it. Thus, each slice has priority to its contracted capacity over other slices.

4) Performance Criteria: In this work, we utilize conventional performance criteria for comparing exact and approximate solutions – the mean squared error.

IV. Machine Learning Algorithms

In this section, we first formalize a model of resource slicing at the air interface aimed at fair priority-based isolation of slices. Then, we proceed to present the exact solution algorithm. Finally, we introduce four supervised learning techniques for speeding up resource arbitration execution.

A. Assumed Resource Arbitration Scheme

We are interested in the demand range in which resources are enough to provide the users with the minimum data rates but insufficient for allocating the maximum data rates to all. This corresponds to population vectors $\mathbf{N} \in \mathbb{N}^S$ such that

$$\mathbf{N}\mathbf{R}^{min} \leq C \leq \mathbf{N}\mathbf{R}^{max}. \tag{3}$$

For such $\mathbf{N}$, a resource allocation that (i) provides max–min fairness to users taking account of slice priorities in terms of whether the contracted number of users is exceeded and by how much, (ii) satisfies the minimum and maximum data rate constraints, and (iii) uses up the whole available resource quantity, can be found as a solution to the problem [5]

$$\text{maximize } U(\mathbf{R}) = \sum_{s\in\mathcal{S}} W_s(N_s) N_s \ln(R_s) \tag{4}$$

$$\text{subject to } \mathbf{N}\mathbf{R} = C \tag{5}$$

$$\text{over } \mathbf{R} \in \mathbb{R}^S : R_s^{min} \leq R_s \leq R_s^{max}. \tag{6}$$

Let the weight function in (4) be given by

$$W_s(N_s) = \begin{cases} 1, & N_s \leq N_s^{cont}, \\ N_s^{cont}/N_s, & N_s > N_s^{cont}, \end{cases} \tag{7}$$

thus ensuring the max–min fair resource allocation to users as long as their number in the corresponding slices does not exceed the contracted quantity. The "violating" slices, in which $N_s > N_s^{cont}$, are penalized in such a way that their resource allocation is calculated for only N_s^{cont} users whenever it is possible without compromising the minimum data rates. The constraint (5) ensures not only that the total allocation does not exceed the available capacity C, but also that all available resources are allocated. Finally, the box constraints (6) guarantee that the minimum and maximum data-rate requirements in slices are satisfied. Let us denote the column vector of weights by $\mathbf{W} = (W_s(N_s))_{s\in\mathcal{S}}$.

The objective function in (4) is differentiable and strictly concave, and the feasible region is compact and convex. Thus, there exists a unique (up to R_s corresponding to $N_s = 0$) maximum for $U(\mathbf{R})$ in the feasible region, which Lagrangian methods can find. In the next subsection we provide an algorithm for finding the exact solution of (4)–(6).

B. Exact Solution

For finding the exact solution of the problem (4)–(6) we propose to employ Algorithm 1. It uses a recursive function, *findCandidates*, which populates the set of solution candidates, $\mathcal{R}$, by considering all possible combinations of active constraints (6). The algorithm operates as follows. The unique solution to the problem (4)–(5), with the box constraints (6) lifted, can be easily found as

$$R_s^{stat} = \frac{W_s C}{\mathbf{N}\mathbf{W}},\ s \in \mathcal{S}. \tag{8}$$

If $\mathbf{R}^{stat}$ satisfies (6) then it is the optimum and $\mathcal{R}$ contains only one element. If, however, $\mathbf{R}^{stat}$ is not feasible then *findCandidates* is run recursively with one additional constraint, $R_s = R_s^{max}$ or $R_s = R_s^{min}$, activated at each call for all $s \in \mathcal{S}$ corresponding to non-zero N_s. If, e.g., at the current call, 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 (4) is chosen among the members of $\mathcal{R}$.

Note that, unfortunately, whenever (8) does not provide a feasible solution, the time complexity of Algorithms 1 is exponential on 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 $O(S^4)$ 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 this reason, we need faster algorithms that can be found in the ML field. Further in this section, we address several such approximations to speed up the resource arbitration process.

Algorithm 1: Exact Solution of (4)–(6)

Input: such $\mathbf{N} \in \mathbb{N}^S$ that satisfies (3), the parameters $\mathbf{R}^{min}$, $\mathbf{R}^{max}$, $\mathbf{N}^{cont}$ and C, the weights $\mathbf{W}$ obtained by (7)
Output: $\mathbf{R}$ solving (4)–(6)
Function FINDCANDIDATES($\mathbf{N}^{cur}$, $\mathbf{R}^{cur}$, C^{cur}, $\hat{s}$):
  if $\mathbf{N}^{cur}\mathbf{R}^{min} \leq C^{cur} \leq \mathbf{N}^{cur}\mathbf{R}^{max}$ **then**
    $\mathbf{R} := \mathbf{R}^{cur}$
    $\mathbf{R}^{stat} := \frac{C^{cur}}{\mathbf{N}^{cur}\mathbf{W}}\mathbf{W}$ // stationary point
    if $R_s^{min} \leq R_s^{stat} \leq R_s^{max}$ $\forall s \in \mathcal{S}$ *such that* $N_s^{cur} > 0$ **then**
      for *such* $s \in \mathcal{S}$ *that* $N_s^{cur} > 0$ **do**
        $R_s := R_s^{stat}$
      $\mathcal{R} := \mathcal{R} \bigcup \{\mathbf{R}\}$ // add candidate
      if $\hat{s} = 1$ **then**
        return
    else
      for *such* $s = \hat{s}, ..., S$ *that* $N_s^{cur} > 0$ **do**
        $\mathbf{N}^{next} := \mathbf{N}^{cur}$
        $\mathbf{R}^{next} := \mathbf{R}^{cur}$
        $N_s^{next} := 0$
        for $R_s^{bound} := R_s^{min}, R_s^{max}$ **do**
          $R_s^{next} := R_s^{bound}$
          $C^{next} := C^{cur} - N_s^{cur} R_s^{bound}$
          FINDCANDIDATES($\mathbf{N}^{next}$, $\mathbf{R}^{next}$, C^{next}, $s+1$)

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

C. ML Approximations

As illustrated in Fig. 1, the resource arbitration scheme has two types of parameters: the SLA thresholds, $\mathbf{R}^{min}$, $\mathbf{R}^{max}$ and $\mathbf{N}^{cont}$, as well as the number of instantiated slices S change rarely (e.g., when a slice is added or removed), whereas the demand parameters $\mathbf{N}$ corresponding to the numbers of ongoing sessions in slices may change on sub-second timescales. Therefore, we can consider prolonged periods of time when the only parameter that varies is $\mathbf{N}$, thus allowing for efficient use of supervised learning techniques. In this work, we investigate four techniques: linear regressions, polynomial regression, the random forest regressor, and the two-layer ANN.

Assume S, $\mathbf{R}^{min}$, $\mathbf{R}^{max}$, $\mathbf{N}^{cont}$ and C constant and consider training data

$$\mathcal{D} = \{(\mathbf{N}^{(1)}, \mathbf{R}^{(1)}), (\mathbf{N}^{(2)}, \mathbf{R}^{(2)}), ..., (\mathbf{N}^{(K)}, \mathbf{R}^{(K)})\}, \quad (9)$$

where, for any sample $k = 1, ..., K$, $\mathbf{N}^{(k)}$ satisfies (3) and $\mathbf{R}^{(k)}$ represents the solution to (4)–(6) for $\mathbf{N} = \mathbf{N}^{(k)}$. Our task is to use $\mathcal{D}$ to build such a vector function f that, for any $\mathbf{N}$ satisfying (3), $\widetilde{\mathbf{R}} = f(\mathbf{N})$ represents a suitable approximation of the solution of (4)–(6) for $\mathbf{N}$.

1) Linear Regression: The baseline technique we use is linear regression. Here, we obtain the vector of user data rate approximations as

$$\widetilde{\mathbf{R}} = (1, N_1, ..., N_S)\mathbf{B}, \quad (10)$$

with the regression coefficients to be determined from the training data $\mathcal{D}$. For this, we employ the method of least squares. Then, the matrix of regression coefficients, $\mathbf{B} = (\beta_{i,s})_{i=\overline{0,S}, s=\overline{1,S}}$ can be computed by the well-known formula

$$\mathbf{B} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y} \quad (11)$$

with

$$\mathbf{X} = \begin{bmatrix} 1 & N_1^{(1)} & \dots & N_S^{(1)} \\ 1 & N_1^{(2)} & \dots & N_S^{(2)} \\ \vdots & & \ddots & \\ 1 & N_1^{(K)} & \dots & N_S^{(K)} \end{bmatrix}, \quad (12)$$

$\mathbf{Y} = [\mathbf{y}_1 \ \dots \ \mathbf{y}_S]$ where $\mathbf{y}_s = (R_s^{(k)})_{k=\overline{1,K}}$ is a column vector.

2) Polynomial Regression: The next technique we are interested in is polynomial regression. In a polynomial regression of degree m the user data rate approximation is computed as

$$\widetilde{\mathbf{R}} = (1, N_1, ..., N_S, N_1^2, ..., N_S^2, ..., N_1^m, ..., N_S^m,)\mathbf{B}, \quad (13)$$

where $\mathbf{B}$ is obtained by (11) in which each row of matrix $\mathbf{X}$ is appended on the right with the powers of $N_s^{(k)}$

$$\left(N_1^{(k)}\right)^2, \dots, \left(N_S^{(k)}\right)^2, \dots, \left(N_1^{(k)}\right)^m, \dots, \left(N_S^{(k)}\right)^m. \quad (14)$$

3) Random Forest Regressor: Another technique that we consider is the random forest (RF) regressor. Random forest is a classification algorithm consisting of multiple decision trees. Based on the characteristics of the training data set, the algorithm builds a decision tree consisting of simple questions (e.g., "Is N_1 greater than 20?"). The decision of splitting is based on a special criterion, which in our case is the mean squared error. Each sample from the data set passing through the branches of the tree (answering all questions) as a result will fall into some data rate group.

4) Two-Layer Artificial Neural Network: The last ML technique investigated is an ANN with one sigmoid-activated hidden layer of size J and a linear activation output. More specifically, the approximation is computed as

$$\begin{aligned} &\widetilde{\mathbf{R}} = (1, h_1, ..., h_J)\mathbf{B}^{(2)}, \\ &h_j = \frac{1}{1 + e^{-\hat{h}_j}}, \ j = 1, ..., J, \\ &\hat{\mathbf{h}} = (1, N_1, ..., N_S)\mathbf{B}^{(1)}. \end{aligned} \quad (15)$$

The weight matrices $\mathbf{B}^{(1)}$ and $\mathbf{B}^{(2)}$ are obtained from $\mathcal{D}$ via the Gradient Descent method using the sum quadratic loss

$$L_k = \sum_{s \in \mathcal{S}} \left(f_s(\mathbf{N}^{(k)}) - R_s^{(k)} \right)^2. \quad (16)$$

V. Numerical Evaluation

In this section, we present our results. We start by suggesting an implementation scenario for ML-enhanced resource arbitration and then proceed providing the parameters and assessing the approximation accuracy.

A. Proposed Implementation Scenario

A proposed scenario of the ML-enhanced resource arbitration consists of two phases shown in Fig. 2. Each time a slice is instantiated, removed or modified, a training phase begins, during which the problem (4)–(6) is solved exactly and the observed system states along with the exact solutions are collected into a training data set. Once the training set is populated, an ML model is trained. Training data can be collected and employed in batches in the case of ANN.

A part of the collected data is reserved for ML model validation. Once a sufficient level of accuracy is achieved on the validation data, the process moves on to the ML use phase. Here the ML technique is used to solve the problem (4)–(6). However, the accuracy is monitored either through periodical comparison with the exact solution or some relevant performance measures. Whenever insufficient accuracy is detected, e.g., due to a change in demand yielding population vectors substantially differing from the training data, the process starts over from the training phase.

B. Accuracy Assessment

The parameters for numerical evaluation are shown in Table I. The overall data set contains $K = 700$ training and 300 test samples. Each $N_s^{(i)}$, $s \in \mathcal{S}$, was sampled from the uniform distribution on $[0, \lfloor C/R_s^{min} \rfloor]$, after which the population vectors $\mathbf{N}^{(i)}$ were validated by checking (3). $\mathbf{R}^{(i)}$ were computed via Algorithm 1. Our dataset contains 300 test samples and 700 training samples. The sample generation is subject to a uniform distribution from 0 to $\lfloor C/R_s^{min} \rfloor$. The dataset contains only those samples, who fulfill the condition (3). To implement the RF regressor we utilized the *RandomForestRegressor* function from the *scikit-learn* library [16]. The ML techniques were compared in terms of average loss (16) over the test data set.

TABLE I
Numerical Evaluation parameters

Parameter	Value
Number of slices, S	3
Total capacity, C (Gbps)	150
Minimum threshold, $\mathbf{R}^{min}$ (Gbps)	$[2, 12, 1]$
Maximum threshold, $\mathbf{R}^{max}$ (Gbps)	$[5, 15, 10]$
Contracted numbers of users, $\mathbf{N}^{cont}$	$[22, 3, 45]$
Training data set size, K	700
Test data set size	300
Polynomial regression: degree	2, 5
RF: number of trees	100
RF: maximum depth of trees	5, None
RF: number of features to select splitting	2
ANN: hidden layer size, J	10
ANN: gradient descent iterations	5×10^5
ANN: learning rate	10^{-3}

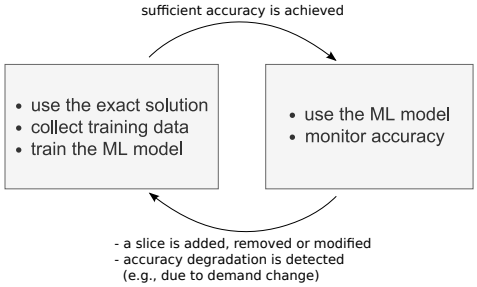


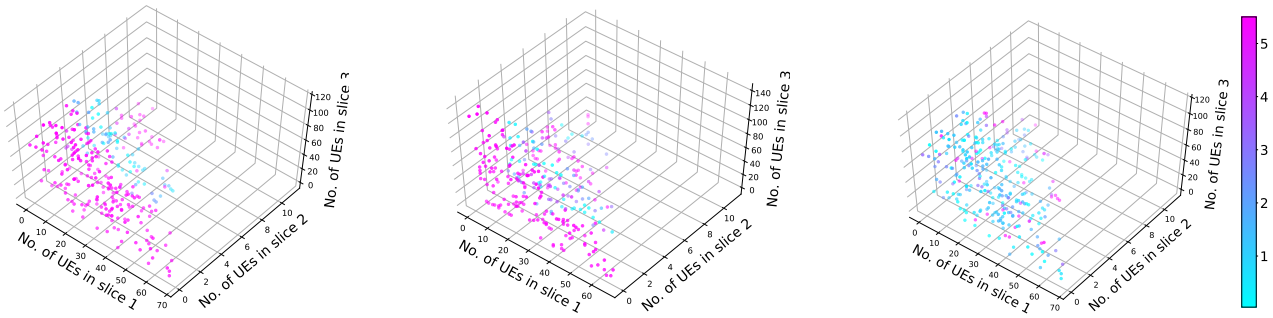
Fig. 2. Considered implementation scenario.

Recalling the need for simple yet efficient algorithms, we start our numerical evaluation by presenting the results for regression models in Fig. 3, where in addition to linear regression, two polynomial regressions of different orders are presented. By analyzing the presented result, one may observe that the linear regression model is characterized by the worst approximation for all the considered ranges of users in slices. Qualitatively similar approximation is produced by the low-order polynomial regression, see Fig. 3(b) for order 2. However, increasing the order of the regression, we may observe significant improvements in terms of accuracy. Note that the computational complexity of polynomial regression does not heavily depend on its order, and the underlying algorithms are relatively lightweight, see Section IV-C, especially compared to the exact solution provided in Section IV-B. Thus, polynomial regressions of high orders can be considered as potential candidates for practical implementation.

We now proceed by comparing the results of the high-order polynomial regression to those of more advanced algorithms such as RF and ANN. To this aim, Fig. 4 demonstrates the performance of RF, RF of depth 5 and two-layer ANN algorithms. By comparing the presented results, one may observe that RF provides accuracy that is comparable to the high-order polynomial regression algorithm. Surprisingly, both RF of depth 5 and the two-layer ANN provide slightly worse results for all considered numbers of sessions in slices. In general, our results imply that for the considered conditions polynomial regression can provide the trade-off between computational complexity and accuracy.

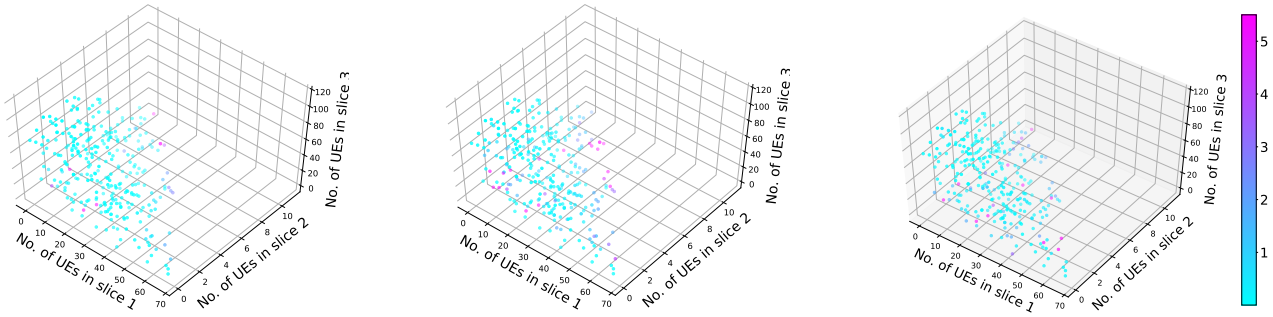
VI. Conclusions

Enabling network slicing for the 5G system requires solution of complex optimization problems that try to simultaneously ensure performance isolation and fairness among slices sharing the air interface. Motivated by this challenge, we implemented several approximate solutions based on ML algorithms in this paper. Aiming for balance between computational complexity and accuracy, the tested algorithms included linear regression, polynomial regression of different orders, the random forest regressor, and the two-layer artificial neural network. Our results demonstrate that a polynomial regression or



(a) Linear Regression (b) Polynomial regression, degree 2 (c) Polynomial regression, degree 5

Fig. 3. The results of linear and polynomial regression algorithms.



(a) Random forest (b) Random forest, depth 5 (c) Two-layer neural network

Fig. 4. The results of random forest and two-layer neural network algorithms.

high order provides the desired balance between computational complexity and accuracy of results. The performance of this algorithm is comparable to those of random forest and neural network and much superior compared to linear regression. The future work includes addressing realitic constraints on the approximation, analysis of the system under a variable number of slices and total capacity C, and use of detailed sample distributions to assess accuracy vs. demand trade-offs.

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 journal on selected areas in communications*, vol. 35, 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 Journal on Selected Areas in Communications*, 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 Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[4] "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/etsi/15.02.00 60/ts 123501v150200p.pdf

[5] N. Yarkina, Y. Gaidamaka, L. M. Correia, and K. Samouylov, "An analytical model for 5G network resource sharing with flexible SLA-oriented slice isolation," *Mathematics*, vol. 8, no. 7, p. 1177, 2020.

[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] 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.

[8] A. A. Al-Khatib and A. Khelil, "Priority-and reservation-based slicing for future vehicular networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 36–42.

[9] Q. Wang, J. Fu, J. Wu, B. Moran, and M. Zukerman, "Energy-efficient priority-based scheduling for wireless network slicing," in *2018 IEEE Global Communications Conference*. IEEE, 2018, pp. 1–6.

[10] H. Ko, J. Lee, and S. Pack, "Priority-based dynamic resource allocation scheme in network slicing," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 62–64.

[11] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing management & prioritization in 5G mobile systems," in *European Wireless 2016; 22th European Wireless Conference*. VDE, 2016, pp. 1–6.

[12] N. An, Y. Kim, J. Park, D.-H. Kwon, and H. Lim, "Slice management for quality of service differentiation in wireless network slicing," *Sensors*, vol. 19, no. 12, p. 2745, 2019.

[13] Y. Sadovaya, D. Moltchanov, H. Nikopour, S.-p. Yeh, W. Mao, O. Orhan, S. Talwar, and S. Andreev, "Self-interference assessment and mitigation in 3GPP IAB deployments," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[14] 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.

[15] 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.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.