Machine Learning based Resource Orchestration for 5G Network Slices

Nazih Salhab*†, Rana Rahim‡, Rami Langar*, Raouf Boutaba§
* University Paris-Est, LIGM-CNRS UMR 8049, UPEM, F-77420, Marne-la-Vallée, France
† LTRM, EDST, Lebanese University, Tripoli, Lebanon
‡ LTRM, Faculty of Science, Lebanese University, Tripoli, Lebanon
§ David R. Cheriton School, University of Waterloo, Waterloo, ON, Canada
E-mails: nazih.salhab@u-pem.fr; rana.rahim@ul.edu.lb; rami.langar@u-pem.fr, rboutaba@uwaterloo.ca

***Abstract*—5G will serve heterogeneous demands in terms of data-rate, reliability, latency, and efficiency. Mobile operators shall be able to serve all of these requirements using shared network infrastructure's resources. To this end, we propose in this paper a framework for resource orchestration for 5G network slices implementing four Quality of Service pillars. Starting from traffic classification, demands are marked so that they are best served by dedicated logical virtual networks called Network Slices (NSs). To optimally serve multiple NSs over the same physical network, we then implement a new dynamic slicing approach of network resources exploiting Machine Learning (ML). Indeed, as demands change dynamically, a mere recursive optimization leading to progressive convergence towards an optimum slice is not sufficient. Consequently, we need an initial well-informed slicing decision of physical resources from a total available resource pool. Moreover, we formalize both admission control and slice scheduler modules as Knapsack problems. Using our 5G experimental prototype based on OpenAirInterface (OAI), we generate a realistic dataset for evaluating ML based approaches as well as two baselines solutions (i.e. static slicing and uninformed random slicing-decisions). Simulation results show that using regression trees as an ML based approach for both classification and prediction, outperform other alternative solutions in terms of prediction accuracy and throughput.**

***Index Terms*—Network Slicing, Machine-Learning, Resource Orchestration, 5G, OAI**

I. Introduction

5G will not only feature increased throughput and higher-mobility that is known as extreme Mobile Broadband (xMBB) but also, it will support Machine to Machine communications. Multiple reports predict exponential inflation of Internet of Things (IoT) communication in its two facets (Massive and Critical), that are known as Massive Machine Type Communications (mMTC) and Ultra-Reliable Low-Latency Communications (URLLC) [1]. Each class (xMBB, mMTC and URLLC) has its own requirements in terms of throughput, mobility, reliability, latency, energy efficiency, and has different characteristics when it comes to connection and traffic densities [1]. Accordingly, network infrastructure has to serve heterogeneous demands. Although demands are heterogeneous, the same ultimate objective is shared, which is to guarantee Quality of Service (QoS).

For cost-effectiveness, mobile network operators shall optimize resources' supply to meet demands. This is especially interesting because of continuously pushing economic reasons due to natural scarcity of resources (spectrum and power) or even infrastructure resources (Compute, Network, and Storage). Leveraging two enablers, namely Software-Defined Networking (SDN) and Network Function Virtualization (NFV), a solution consists of provisioning dedicated logical networks, also known as Network Slices (NSs) [2]. To ensure QoS, the first pillar consists of classifying and marking traffic based on QoS Class Identifiers (QCIs) [3], so that each type of traffic is associated to an adequate architecture implemented by a tailor-made NS. The former "one-size-fits-all" scheme shall be replaced by fine-tuned slices that are optimized to secure objectives, while meeting stipulated constraints. To provide End-To-End (E2E) QoS, four QoS pillars need to be implemented [3].

After classification and marking, we consider a case of several NSs sharing an initial resource allocation from a unique resource pool that is typically imposed by a particular radio design. Accordingly, multiple slices with different resources' percentage from the total available resource pool (slicing ratio) can be observed. However, as demands change with time, a dynamic optimization is necessary. On top of this, initial well-informed slicing ratios are needed. To answer the question of how to get such optimal ratio, we propose, in this paper, employing Machine Learning (ML) and particularly, Regression Trees (RTs) to do traffic trend forecasting and throughput maximization. An RT is a graph that uses branching method to illustrate every possible outcome of a decision. Specifically, we use multiple predictor variables extracted from the traffic profile (such as, time, day-of-the-week, planned-event existence, and other environmental factors) to predict an optimal value of bandwidth ratio. We use our 5G experimental prototype [4, 5] based on OpenAirInterface™ (OAI) to generate a realistic dataset and benchmark the performance of several ML based approaches as well as two baselines solutions: static slicing and un-informed random slicing-decisions. Our contributions can be thus summarized as follows:

- We design a framework consisting of several building blocks integrating four QoS pillars for resource orchestration in 5G network slices.
- We provide formulation, mathematical models and an

algorithm to implement these building blocks using i) ML based approaches for "Classification and Marking" along with ii) Predictions using RTs, iii) Knapsack problems for admission control and scheduling optimization and iv) Agile elasticity algorithms for resource provisioning.

- We show the effectiveness of our proposal compared to other alternative solutions using our 5G experimental prototype based on OAI.

The remainder of this paper is organized as follows. In section II, we present an overview of related work. Section III describes our system design and details our proposed models for implementing the building blocks of our framework. Section IV presents the performance evaluation starting by describing our prototype, used dataset, and ending by simulation results discussion. We finally conclude this paper in section V.

II. Related Work

In this section, we discuss a selection of relevant approaches for radio resource management in 5G networks.

Authors in [6] detailed a framework on the configuration of Radio Resource Management (RRM) in a sliced Radio Access Network (RAN) using static slicing ratios and evaluated the blocking rate and the throughput per Data Radio Bearer (DRB) of different types of slices. However, to achieve efficient resource allocation, we need a dynamic slicing ratio based on traffic load predictions. Authors in [7] used feature-selection based prioritization to predict mobile traffic and they used data from an open dataset used in a big-data challenge. They compared their approach to random-based predictions.

Authors in [8] conceptualized three key building blocks for network slicing. Precisely, they designed a forecasting module, an admission control agent and a scheduler. They used Holt-Winters (HW) method to do traffic prediction. Little details were provided on used dataset and its characteristics. They sacrificed some violation of Service Level Agreement (SLA) for an increase of resource utilization. Differently from them, we included in our design an elastic "Resource Manager" to avoid SLA violation to sustain isolation principle in NS.

Authors in [9] used Seasonal Auto-Regressive Moving-Average (SARIMA) to do predictions. Although, SARIMA is not complex as it does not rely on predictor variables, it fails when there is an unusual growth or slowdown in the time series, unlike predictor-based approaches where predictor variables reflect such unusual change.

Authors in [10] used a stochastic model to represent time series data using Markov Chains (MCs). Although MCs are stochastic processes, but they lack any "memory" by design.

Unlike the aforementioned approaches (HW, SARIMA and MC), we propose an ML based framework for resource orchestration in 5G networks. ML based approaches are expected to be competitive in accuracy due to data abundance that is exploited in continuous training and re-training of models. In addition, differently from such related works, to prove the effectiveness of our proposal, we generated a realistic dataset using our 5G experimental prototype presented in [4, 5].

Finally, authors in [11], [12], [13] used Machine-Learning approaches, namely Random Forests, SVM and GPR, respectively in different contexts. We compare our approach with these ones and discuss the associated results.

III. System Design

Our Network Slices' Orchestrator, shown in Fig. 1, is built upon the concept of resource sharing elaborated in 3GPP technical specification detailing the concept and requirements for network sharing and management architecture [14].

As depicted in Fig. 1, after classification and marking, traffic trends forecasting is introduced to predict the ratio of slices and maximize system's throughput, accordingly. The starting point is getting NSs requirements blueprints that encompass network characteristics (spectral efficiency, latency, reliability, and energy efficiency) required by a service instance [15]. Our designed "Gatekeeper" building block is used to implement two fundamentals roles that are "Classification and Marking" based on received blueprints in order to form the first QoS pillar [3]. We propose to use supervised ML for implementing the classification [16]. Initial data labeling is done according to the details found in the blueprints, which include the SLA.

The "Decision Maker" building block is composed of two sub-modules that are "Forecast Aware Slicer" using ML based regression and an "Admission Controller" that either grants or denies resource's requests according to current and predicted loads. The "Admission Controller" makes decision according to preset policies. Two outcomes are derived from this sub-module. Granted requests are forwarded to the "Slice Scheduler" so that they are served in the nearest time window. This latter implements the third QoS pillar. If admission is not possible, denied requests are forwarded to the "Resource Manager" implementing the fourth QoS pillar on how to proceed in case of high resource's utilization. An Auto-scaler is an example of an agile "Resource Manager" that revokes denied requests by dynamically instantiating, whenever possible, additional workers to serve these requests. It is worth noting that the "Slice Scheduler" plays an interesting role in providing feedback to the "Decision Maker" to reinforce the learning experience.

In what follows, we present in details the aforementioned building blocks by formulating the problem of each block and describing how to solve it.

A. Gatekeeper

To do traffic forecasting based on trends, an initial phase of classification and marking is needed so that aggregation of traffic per slice takes place. Indeed, each use case requires a network slice that is tailor-made to fit its particular service requirements. Table I resumes our assumptions on some QoS Classes of traffic according to SLAs blueprints [17]. As an example, for $l = 3$, QCI = 65 with Guaranteed Bit Rate (GBR) resource type, delay budget of 10 ms, packet loss tolerance of 10^{-2}, and a priority of 0.7, it is suitable for the Mission Critical Push To Talk (MCPTT) service [17]. We assume that every instance (or event) of a point process ξ can be represented by

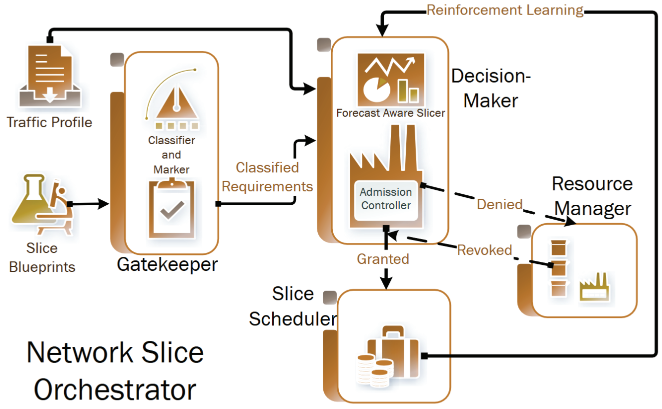

Fig. 1: Block Diagram of 5G Network Slice Orchestrator

$\xi_i^{(l)} = \sum_{t=0}^{T} \delta_t r_i^{(l)}(t)$, where δ_t denotes the Dirac measure for time sample t. Variable $r_i^{(l)}(t)$ denotes requests of tenant i for a traffic class l over the time t.

B. Decision Maker

1) Forecast Aware Slicer: To predict slices' ratios, we use multiple predictor variables, extracted from the traffic profile, that are level, trend, seasonality, time, day-of-the-week, planned-event existence, and cloudy-conditions. Note that for the sake of simplicity, predictors variables are binarized where possible. Cloudy-conditions are fetched through an on-line Application Programming Interface (API) made available by DarkSky [18].

2) Problem Formulation: We formulate our regression model as follows.

$$y = f(X, \beta) + e \tag{1}$$

where $y \in \mathbb{R}$ is a continuous valued scalar output denoting the response variable; e is an independent random noise involved in the statistical relationship between response variable y and predictor variables x_i instead of a perfect deterministic relation; $X \equiv (x_1, x_2, \dots)$ is a vector of predictors variables. Parameters vector $\beta \equiv (\beta_1, \beta_2, \dots)$ is unknown and its variables are evaluated during the simulation based on the chosen regression model. Denoting by c a particular leaf, and by T our tree, our proposed objective function for tree growing, expressed in (2), consists of minimizing the sum S of squared errors of our Tree T [19]:

$$S = \sum_{c \in leaves(T)} \sum_{i \in c} (y_i - m_c)^2 \tag{2}$$

where $m_c = \frac{1}{n_c} \sum_{i \in c} y_i$, the prediction for leaf c with n_c points in it.

3) Proposed Algorithm for growing regression trees: We propose the following simple yet efficient algorithm (elaborated in Algorithm 1) for growing regression trees by progressive elaboration. Starting initially with all nodes and based on iterations that end according to some predefined criteria, we split points to different branches. To express these criteria, we define a parameter q denoting the minimum leaf size for the sought tree. We also define parameter ε as the minimum change threshold for continuing with iterations.

Algorithm 1: ML-based Regression Tree Growing

```
Data: Records of predictors and the optimum response
Result: Grown Tree for responses
do
    Initialize a single node containing all points;
    Calculate $m_c$ and $S$;
    if $\forall$ points in node, predictors are same then
        stop;
    else
        search over all binary splits of all variables for
          the one which reduces S
    end
    if ($Max(\Delta(S)) < \varepsilon$ or $\exists$ cardinal(node) $< q$) then
        stop;
    else
        Take that split and create 2 nodes
    end
while no more new nodes;
```

C. Admission Controller

An "Admission Controller" receives requests that need to be scheduled. Based on the current system load and supported by data generated by the "Forecast Aware Slicer", it decides whether to grant or deny each individual request. In case of denial, it forwards the request to the "Resource Manager" so that a possible resource pool re-dimensioning takes place in order to decrease the chances of service denial as possible, depending on the capabilities of underlying physical system or whenever additional pooled infrastructure resources are available.

Problem Formulation and Resolution: Based on our previous work in [20], we formalize the "Admission Control" problem as D-dimensional Multiple-Choice Knapsack problem that is constrained by multiple limits stipulated by each slice type. The problem can be formalized as follows.

$$\max_{x} \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} v_j x_{ij} \tag{3a}$$

s.t.

$$\sum_{j=1}^{n} w_j^{(d)} x_{ij} \le C_i^{(d)}, \; d \in \{1, \dots, D\}, \; i \in \{1, \dots, m\} \tag{3b}$$

$$\sum_{i=1}^{m} x_{ij} \le 1, \qquad j \in \{1, \dots, n\} \tag{3c}$$

$$x_{ij} \in \{0, 1\}, \qquad i \in \{1, \dots, m\}, \; j \in \{1, \dots, n\} \tag{3d}$$

The objective function in (3a) aims to maximize the value of network resources utilization v, while serving network slices requests as per contracted SLA. The set of D constraints (3b) specifies the stipulations on computing resources among the required demands in terms of Virtual Central Processing Units (vCPUs), and virtual Memory. These resources cannot bypass an upper bound as imposed by underlying system capabilities denoted by C_i^d. Assuming no duplication, constraint (3c) enforces exclusivity, so that each resource request is served by at most one slice.

The solution of this problem is a set of x_{ij} that are binary values indicating whether a request j is served by slice i and thus admitted into the system or not. Index variables n and m denote the number of requests and the number of slices, respectively. Each admitted request j is valued as v_j.

To solve this NP-hard problem, we can use our polynomial time heuristic algorithm proposed in our previous work in [20].

D. Slice Scheduler

Once "Admission Controller" has warranted grants to a network slice, such demand is adressed to the "Slice Scheduler" to properly serve its corresponding traffic with minimal time duration. We denote by p_j the processing time of a transmission request j, out of n requests, such that its timespan is $c_{j,t}$. Our problem consists in finding a schedule minimizing the total time duration. We define a binary decision variable x_{jt} to indicate if a request j is scheduled in time period t. Our slice scheduler can be formalized as an optimization problem expressed as follows.

$$\min_{x} \quad z = \sum_{j=1}^{n} \sum_{t=1}^{T} c_{jt} x_{jt} \tag{4a}$$

s.t.

$$\sum_{j=1}^{n} \sum_{s=max\{0;t-p_j+1\}}^{t} x_{js} \leq N, \; t \in \{1, \dots, T\} \tag{4b}$$

$$\sum_{t=1}^{T} x_{jt} \leq 1, \qquad j \in \{1, \dots, n\} \tag{4c}$$

$$x_{jt} \in \{0, 1\}, \;\; j \in \{1, \dots, n\}, \;\; t \in \{1, \dots, T\} \tag{4d}$$

Knowing that the host implementing a slice has finite capacity and can handle maximum up to N requests concurrently, constraint (4b) stipulates that during time period t, N requests can be executed, at most. Constraint (4c) means that each request has to be scheduled only once. Finally, constraint (4d) stipulates that each request should be either served at time period t or deferred.

It is worth noting that the formulated problem in (4) is NP-hard and corresponds to a Knapsack problem, which is a particular case of a D-dimensional Multiple-Choice Knapsack problem. A general heuristic has been proposed in our previous work in [20] and can be thus used as well to solve such particular case in a polynomial time.

E. Resource Manager

Our "Resource Manager" exploits the usage of microservices to auto-scale infrastructure resources and thus provides elastic resources to decrease requests' rejection ratio.

According to predictions made by the "Forecast Aware Slicer", the "Admission Controller" sub-module of "Decision Maker" might request some additional elastic resources from "Resource Manager" so that it does not violate the SLAs. Such observation is sent back by the "Slice Scheduler" to the "Decision Maker" as feedback. Reinforcement Learning (RL) can be used here to enable the "Decision Maker" acting as an agent to learn how to perform its decisions for elastic resources, while interacting with an unknown changing environment. Indeed, RL aims to maximize the cumulative reward autonomously [8]. As predictions might be wrong, the "Decision Maker" agent can learn the optimal behavior through repeated trial-and-error interactions with the environment.

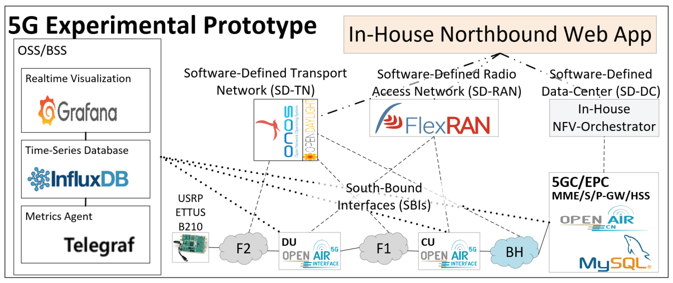
5G Experimental Prototype
In-House Northbound Web App
OSS/BSS
Realtime Visualization
Grafana
Time-Series Database
InfluxDB
Metrics Agent
Telegraf
Software-Defined Transport Network (SD-TN)
ONOS
OPEN DAYLIGHT
Software-Defined Radio Access Network (SD-RAN)
FlexRAN
Software-Defined Data-Center (SD-DC)
In-House NFV-Orchestrator
South-Bound Interfaces (SBIs)
USRP ETTUS B210
F2
DU
OPEN AIR INTERFACE 5G
F1
CU
OPEN AIR INTERFACE 5G
BH
5GC/EPC
MME/S/P-GW/HSS
OPEN AIR CN
MySQL

Fig. 2: Our 5G experimental prototype block diagram [5]

IV. Performance Evaluation

In this section, we evaluate the performance of our proposed network slice orchestrator framework using our 5G experimental prototype. We start by presenting our prototype. Then, we present our methodology to generate the dataset used in our simulations, followed by a presentation of the obtained results.

A. 5G Experimental Prototype Overview

Fig. 2 depicts our 5G experimental prototype based on OAI. Top rectangle (in Orange) hosts our proposed orchestrator and its building blocks. It interacts with a Software-Defined RAN, namely, FlexRAN [21] through a northbound interface to manage sub-jacent RAN nodes implemented using OAI. It interfaces with open-source Software-Defined Controllers for both Transport Network (TN) and Data centers (DC). We implement Operation Support Subsystem/Business Support Subsystem (OSS/BSS) using open-source software to collect infrastructure metrics (Telegraf), store it in time-series database (InfluxDB) and visualize it using a dashboard (Grafana) [2].

B. Dataset Generation and Simulation Environment

A dataset is generated using our 5G experimental prototype by running, for 24 hours, a background script interacting with FlexRAN to collect the Medium Access Control (MAC) layer statistics of our Commercial Off-The Shelf (COTS) User Equipments (UEs) and IoT devices (i.e., temperature and humidity sensors, light and motion sensors, Intelligent Switch and Smart lamp). The collected statistics include provisioned slicing ratios, priority, QCI, power measurements, among others. With background processes requiring internet connection, UEs are periodically moving in our laboratory space to change radio conditions. IoT devices' traffic is aggregated through two

TABLE I: Standardized QCI characteristics [17]

l	QCI	Resource Type	Delay Budget	Loss Rate	Priority
0	1	GBR	100 ms	10^{-2}	2
1	3	GBR	50 ms	10^{-3}	3
2	6	non-GBR	300 ms	10^{-6}	6
3	65	GBR	10 ms	10^{-2}	0.7
4	66	GBR	100 ms	10^{-2}	2

TABLE II: Simulation Parameters

Parameter	Value	Parameter	Value
$C^{(1)}$	2 vCPUs	q (Complex)	36
$C^{(2)}$	1 GB RAM	q (Medium)	12
		q (Simple)	4
D (Knapsack)	2	q (ensemble)	8
# Slices	3	N (PRBs)	100
# UEs.	3	Bandwidth	20 MHz
# IoT Devices	6	ε	0.05
S (Complex)	160	K (Fine KNN)	1
S (Medium)	20	K (Coarse KNN)	100
S (Simple)	4	K (Other KNN)	10

IoT Gateways implemented on Raspberry Pis and sent to the Cloud using our 5G prototype. ML models are implemented using MATLAB® on Dual Intel Core i7, 2.4 GHz, 4-Cores 7th Gen. with 16 GB of RAM. Afterwards, RTs model is compiled as standalone application for Linux so that it is re-used in our 5G experimentation prototype. Several ML based models are implemented to classify different blueprints (QCI, Resource Type, Loss Rate, Priority Level) corresponding to three conventional types of 5G slices: xMBB, URLLC and mMTC, as summarized in Table I. Simulation parameters are summarized in Table II. Note that we used 20 MHz channel bandwidth for our Universal Software Radio Peripheral (USRP) B210 [22].

C. Simulation Results

Table III reports the prediction's accuracy, speed and training time of different classification models of 150 tenants' requests with corresponding blueprints. We can notice that the majority of model types provides an accuracy of more than 90% except for coarse K-Nearest Neighbor (KNN), cosine KNN, boosted trees and RUSBoosted trees. Linear Discriminant provides the highest accuracy.

Table IV reports different performance metrics including the Root Mean Square Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), and coefficient of determination (R^2) [19] for the used prediction models. We can see that trees (complex, medium, simple and even boosted trees) perform the best. However, this comes at the expense of an increased training time, as shown in Fig. 3. Indeed, from that figure, we can clearly see that, although RTs provide the lowest RMSE and highest prediction speed, their training time is high compared to the others. However, this can be acceptable since the training's need is not as frequent as the prediction's need.

To further show the benefit of ML-based RTs, we compare in Fig. 4, the predicted values of slicing ratios with the Optimum, Static and Random-slicing approaches. The boxplots of average gap to the optimum values are shown in Fig. 5. Note that Optimum values are calculated using a bottom-up

TABLE III: Classification Benchmarking

Class	Classification Model	Prediction Accuracy	Prediction Speed (obs/sec)	Train. Time (sec)
Trees	Complex tree	94.7	6000	1.1487
	Medium Tree	94.7	5800	0.7825
	Simple Tree	95.3	9200	0.6463
KNN	Fine KNN	94.7	2200	1.6188
	Medium KNN	94.7	2000	1.5302
	Coarse KNN	64.7	3600	1.4441
	Cosine KNN	84.7	2700	1.7042
	Cubic KNN	94	3500	1.6235
	Weighted KNN	95.3	4400	1.8926
SVM [12]	Linear	96.7	1700	3.234
	Quadratic	96	1900	2.941
	Cubic	94.7	2800	3.8789
	Fine Gaussian	92	2800	3.792
	Medium Gaussian	96.7	2900	3.6911
	Coarse Gaussian	95.3	2500	3.5455
Ensemble [11]	Boosted Trees	33.3	3100	2.1866
	Bagged Trees	94	480	4.7621
	Subspace Discrim.	95.3	410	6.8598
	Subspace KNN	93.3	320	7.4715
	RUSBoosted Tree	33.3	8200	6.4955
Discriminant	Linear	98	6200	1.2942
	Quadratic	96.7	3700	1.9226

TABLE IV: Prediction Benchmarking

Class	Regression	RMSE	R^2	MSE	MAE
Linear	Basic	28.77	0.16	827.66	25.61
	Interactions Linear	29.34	0.12	860.91	25.52
	Robust Linear	29.1	0.14	846.59	24
	Stepwise Linear	29.41	0.12	864.82	25.82
Trees	Complex Tree	5	0.97	25.03	3.07
	Medium Tree	5.17	0.97	26.77	3.15
	Simple Tree	5.76	0.97	33.18	3.77
SVM [12]	Linear	31.48	0.01	990.92	21.74
	Quadratic	18.21	0.66	331.7	14.25
	Cubic	16.82	0.71	282.79	13.4
	Fine Gaussian	23.69	0.43	561.09	19.51
	Medium Gaussian	17.48	0.69	305.47	14.12
	Coarse Gaussian	29.06	0.14	844.43	20.32
Ensemble [11]	Boosted Trees	5.41	0.97	29.23	3.24
	Bagged Trees	12.52	0.84	156.74	10.13
GPR [13]	Squared Exponential	16.87	0.71	284.54	13.62
	Matern 5/2	17.05	0.7	290.66	13.73
	Exponential	17.91	0.67	320.81	13.6
	Rational Quadratic	16.87	0.71	284.54	13.62

estimation by aggregating demands of each slice and deducing the ratios. For the static approach, we assume a ratio of 50% for the xMBB slice. We repeated this benchmarking using 1000 observations. We can see that the random approach performs the worst with an average gap of 30% to the optimal slicing ratio. On the other hand, ML-based RTs outperforms both static and random approaches with an average gap of 5% only to the optimal approach. This is related to its prediction accuracy, which is the highest among the remaining schemes.

Finally, we compare our RTs based "Forecast Aware Slicer" with a realistic case where no traffic profile information exist or where such sub-module is left-out for complexity reduction. We ran our prototype in two modes by activating and then deactivating the "Forecast Aware Slicer" sub-module. Fig. 6 depicts the obtained throughput for the xMBB Slice during five hours of simulation. Interestingly, we can see that the forecasting process allows to increase the throughput by

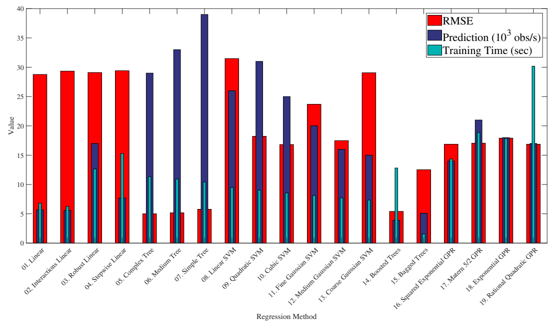

Fig. 3: Performance comparison of Regression Models

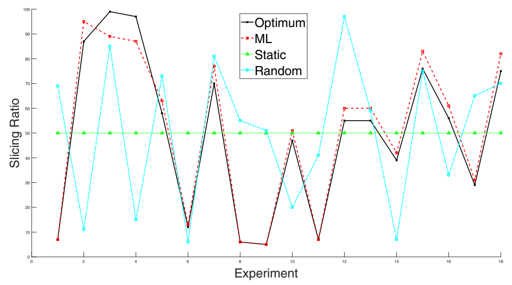

Fig. 4: Ratio of Optimum, ML, Static, Random

approximately 30% compared to the case where the "Forecast Aware Slicer" sub-module is deactivated.

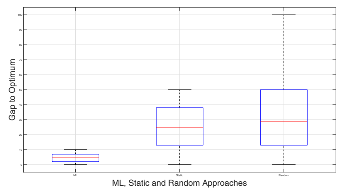

Fig. 5: Boxplots of ML, Static, Random Gap to Optimum

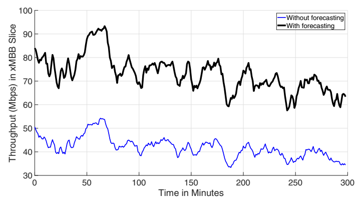

Fig. 6: xMBB Slice throughput with/without forecasting

V. Conclusion

In this paper, we have presented a novel framework based on Machine-Learning (ML) to address the network slice orchestration in 5G networks. Specifically, we have designed and implemented four building blocks, namely, "Gatekeeper" for classification and marking, "Decision Maker" with "Forecast Aware Slicer" and "Admission Controller" sub-modules, "Slice Scheduler", and "Resource Manager", in a 5G prototype using OAI. From our experiments, we have observed that Regression Trees (RTs) outperform other ML models in term of prediction accuracy. In particular, compared to linear based regressions, RMSE is divided by six, prediction speed almost quadrupled but training time has slightly increased. We also showed that the average gap between RTs and the Optimum approach is only 5%. In addition, RTs allow to increase the throughput by approximately 30% compared to the case where the forecasting process is disabled.

In the future, we plan to further exploit ML techniques using deep reinforcement learning and evaluate the performance of our "Decision Maker" building block using neural networks.

Acknowledgement

This work was partially supported by FUI SCORPION project (Grant no. 17/00464), CNRS PRESS project (Grant no. 239953), "Azm & Saade", and the Lebanese University.

References

[1] "3GPP TR 22.864: Feasibility Study on New Services and Markets Technology Enablers - Network Operation; Stage 1 (R.15)," Sept. 2016.
[2] N. Salhab, S. El Falou, R. Rahim, S. E. El Ayoubi, and R. Langar, "Optimization of the implementation of network slicing in 5G RAN," in *IEEE MENACOMM*, Jun. 2018.
[3] J. Kurose and K. Ross, *Computer networks: A top down approach*, 2010.
[4] N. Salhab, R. Rahim, and R. Langar, "NFV Orchestration Platform for 5G over On-the-fly provisioned Infrastructure," in *IEEE International Conference on Computer Communications (INFOCOM), Paris*, 2019.
[5] "Software Defined Radio Laboratory," https://sdr-lab.u-pem.fr/.
[6] J. Prez-Romero, O. Sallent, R. Ferrs, and R. Agust, "On the configuration of radio resource management in sliced RAN," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018.
[7] Y. Yamada, R. Shinkuma, T. Sato, and E. Oki, "Feature-selection based data prioritization in traffic prediction using machine learning," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018.
[8] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE Conference on Computer Communications (INFOCOM)*, May 2017.
[9] Y. Yu, J. Wang, M. Song, and J. Song, "Network traffic prediction and result analysis based on seasonal arima and correlation coefficient," in *2010 International Conference on Intelligent System Design and Engineering Application*, Oct. 2010.
[10] Y. Xie, J. Hu, Y. Xiang, S. Yu, S. Tang, and Y. Wang, "Modeling oscillation behavior of network traffic," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, Sep. 2013.
[11] H. Xu, Y. Pan, J. Li, L. Nie, and X. Xu, "Activity recognition method for home-based elderly care service based on random forest and activity similarity," *IEEE Access*, vol. 7, Jan. 2019.
[12] A. Y. Nikravesh, S. A. Ajila, C. Lung, and W. Ding, "Mobile network traffic prediction using mlp, mlpwd, and svm," in *2016 IEEE International Congress on Big Data (BigData Congress)*, Jun. 2016.
[13] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless traffic prediction with scalable gaussian process," *IEEE Journal on Selected Areas in Communications (JSAC)*, Mar. 2019.
[14] "3GPP TS 32.130: Network Sharing: Concepts (Rel. 14)," Dec. 2016.
[15] "3GPP TR 28.801: Study on management and orchestration of network slicing for next generation network (Release 15)," Jan. 2018.
[16] E. Brynjolfsson and T. Mitchell, "What can machine learning do? workforce implications," *Science*, vol. 358, no. 6370, 2017.
[17] "3GPP TS 23.203: Policy and charging architecture (Rel. 7)," Jun. 2016.
[18] "DarkSky," https://darksky.net/dev/.
[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Chapman and Hall by CRC, 1984.
[20] N. Salhab, R. Rahim, and R. Langar, "Throughput-aware RRHs clustering in cloud radio access networks," in *Global Information Infrastructure and Networking Symposium (GIIS)*, Oct. 2018.
[21] "Flexible Radio Access Network," https://mosaic-5g.io/flexran/.
[22] "Universal Software Radio Peripheral (USRP) (B210)," http://www.ettus.com/all-products/UB210-KIT/.