DRL-Based Dynamic MAC Scheduler Reconfiguration in O-RAN

DRL-based Dynamic MAC Scheduler Reconfiguration in O-RAN

N. Villegas*, J. L. Herrera†, L. Diez*, D. Scotece‡, L. Foschini‡ and R. Agüero*
* Universidad de Cantabria, Spain. e-mail: villegasn@unican.es, {ldiez, ramon}@tlmat.unican.es
† Universidad de Extremadura, Spain. e-mail: jlherrerag@unex.es
‡ Università di Bologna, Italy. e-mail: domenico.scotece, luca.foschini@unibo.it

***Abstract*—This paper introduces reconfiguration mechanisms for a 5th Generation (5G) Medium Access Control (MAC) scheduler, which uses Deep Reinforcement Learning (DRL) and is deployed as an *xApp* within the RAN Intelligent Controller (RIC) framework. The objective is to optimize the allocation of radio resources in 5G networks, seeking to meet QoS requirements while minimizing resource consumption. To this end, we propose the dynamic adjustment of configurable parameters in a Lyapunov-based scheduler, which addresses the challenges posed by the highly dynamic network environment, and the need to meet multiple Quality of Service (QoS) objectives. Exploiting the adaptability of DRL, our solution can effectively respond to fluctuating network conditions, thereby continuously enhancing scheduling decisions in real time. The proposal is evaluated through comprehensive simulations conducted over ns-3 5G-LENA, which demonstrate notable improvements in QoS performance and resource efficiency. The observed results evince the great potential of exploiting DRL to optimize the MAC scheduling in modern wireless communication systems, offering a scalable and adaptive solution for 5G and future 6G networks.**

***Index Terms*—5G, MAC, RIC, QoS, DRL, Lyapunov, ns-3**

I. Introduction

Historically, Radio Access Networks (RANs) have been deployed using a closed, black-box model, with proprietary hardware, software, and interfaces [1]. With the emergence of 5G networks and the widespread adoption of 5G New Radio, RANs have begun to adopt a more open architecture. Specifically, the Open Radio Access Network (O-RAN) Alliance started an effort to develop a new RAN architecture following a non-proprietary approach [2]. The basic idea of O-RAN is to use standardized and open interfaces for all communications between hardware and software components. This ensures interoperability across different vendors.

O-RAN introduces two different architectures: i) one closer to real-time tasks, named Near-Real Time RAN Intelligent Controller (Near-Real Time RIC); and ii) with more loose time constraints, named Non-Real Time RAN Intelligent Controller (Non-Real Time RIC). The RIC is a set of software elements that interact with the RAN to gather information and send control commands [3]. A primary feature of RICs is their ability to allow developers to create services or applications that interact directly with the RAN. In particular, near-real time RAN applications are the so-called *xApps*, while applications close to non-real time tasks are named *rApps*.

Within the access elements, one key mechanism to ensure appropriate QoS levels is MAC scheduling, which determines the distribution of radio resources among different traffic flows. While 5G New Radio (NR) provides the necessary architectural support, it leaves the design of the particular MAC scheduling algorithms open. Thus, deploying effective schedulers that align with the new QoS management framework introduced by 5G is of paramount importance.

A fundamental element of the 5G QoS framework is the concept of QoS flows, each given a 5G QoS Identifier (5QI). The 5QI defines the QoS requirements for each traffic flow, encompassing parameters such as latency, reliability, priority, and allowing the definition of a Guaranteed Flow Bit Rate (GFBR). The categorization of traffic into a number of flows with distinct QoS characteristics enables more granular control over resource allocation in 5G NR. For instance, flows with a higher 5QI priority may be allocated additional resources to meet strict latency or reliability requirements, which are crucial for applications such as eXtended Reality (XR). This differentiation enables the network to adapt its configuration to appropriately tackle varying demands and service requirements.

Nevertheless, attaining this degree of QoS consistency would require a MAC scheduler, that oversees resource distribution in real time, to balance the necessities of highly heterogeneous flows. It is therefore critical to develop schedulers that align with the QoS framework, in order to realize the full potential of 5G and so ensure that diverse applications and services receive appropriate resource prioritization.

This work presents Reinforcement-Based Intelligent Scheduling (RBIS), a MAC scheduling solution for 5G O-RAN, based on the combination of DRL and Lyapunov optimization. RBIS combines the capacity of Lyapunov optimization theory to address a dynamic optimization of the MAC scheduling process, with the potential of DRL to generalize and adapt its behavior to the current scenario status, by learning from its actions.

The contributions of this paper are:

- We present a novel approach for 5G MAC scheduling that combines DRL with Control Theory.
- We integrate the proposed solution within the O-RAN architecture, where the DRL agent is deployed as a *xApp* at the RIC.

- We show that the proposed scheme is able to guarantee a minimum data rate, while optimizing the utilization of allocated resources.
- We validate our approach by means of an extensive simulation campaign over ns-3 5G-LENA.

The rest of the paper is structured as follows. Section II discusses related works on MAC scheduling and QoS management for 5G networks, pointing out how our solution goes beyond them. The system model is described in Section III. Section IV introduces RBIS, our proposed scheduler, which exploits DRL to refine its operation. Its behavior is discussed in Section V, which also validates the correct operation of our solution, by means of an extensive experiment campaign over realistic setups, based on ns3 5G-LENA. Finally, Section VI concludes the paper and provides an outlook of our future work.

II. Related works

The authors in [4] present a thorough survey of the most widespread MAC schedulers in the literature. They highlight that one of the main limitations of the widely implemented Round Robin (RR) algorithm is its rigid approach to fairness. To address this limitation, several channel-aware scheduling strategies have emerged. For example, the Maximum Rate (MR) scheduler fosters users with the highest Modulation and Coding Scheme (MCS), while the best Channel Quality Indicator (CQI) scheduler allocates resources based on channel quality. On the other hand, the Proportional Fairness (PF) algorithm was proposed to better balance the resource distribution. However, these schemes lead to either increased spectral efficiency or resource consumption. The latest proposals have paid attention to meeting specific QoS requirements.

Recently, we have seen an increasing interest in applying Artificial Intelligence (AI) for 5G NR. The authors in [5] present a framework combining Lyapunov optimization and DRL for online computation offloading in mobile-edge computing networks. Furthermore, [6] proposes a Reinforcement Learning (RL) scheduler able to select various resource allocation policies to minimize latency and packet drop rate and so satisfy QoS requirements. The authors conclude that the application of a single scheduling rule is ineffective in meeting the objectives of dynamic network conditions and heterogeneous traffic types.

Recent research on AI focuses on techniques for resource management at higher layers, where systems can tolerate greater latency in decision-making processes [7]. The application of these algorithms for lower-level resource management and slot-level scheduling poses significant challenges, due to timing constraints and the limited or outdated feedback. For instance, [8] introduced a random forest model to predict user modulation and MCS, effectively removing the reliance on the Channel State Indicator (CSI) feedback scheme. In another study, [9] used an Machine Learning (ML) algorithm to forecast buffer occupancy and CQI values in the MAC scheduler, based on historical data. This work proposed leveraging one-time resource allocation for a certain time interval through Prediction Based Scheduling (PBS), to save Physical Downlink Control Channel (PDCCH) resources and reduce resource allocation time. However, this approach operates under the assumption that the channel remains stable, without significant fluctuations, during the allocated periods. Additionally, traditional RL methods, such as Q-learning and Policy Gradient, are often limited to situations with small discrete action spaces. As the number of active users increases, an optimization scheme using a discrete action space becomes increasingly complex and challenging [10].

To the best of our knowledge, no other work has specifically addressed the online configuration of a MAC scheduler based on Lyapunov optimization in 5G/6G networks. Contributions in this field remain scarce, with the application of DRL typically restricted to the selection of alternative scheduling rules or to address sub-problems within a predefined scheduling policy. This limited focus overlooks the potential advantages of integrating DRL with dynamic configuration schemes, which would lead to more responsive and efficient scheduling decisions. In this context, the DRL algorithm has the required time to make decisions, as these are made in sufficiently long windows, which are tailored to the scheduler and aligned with specific QoS requirements. Therefore, time constraint challenges can be effectively addressed by enabling the scheduler to be the responsible of handling the most demanding real-time decisions.

III. System model

In the following we will focus on the downlink scheduling, but a similar approach can be applied to the uplink. Let $\mathcal{N}$ represent the set of Radio Link Control (RLC) queues. A QoS flow is mapped to a corresponding RLC entity (or queue), which ensures the QoS parameters are respected. In this sense, we define Γ_n as the GFBR value of the n^{th} flow. In this context, Drift-Plus-Penalty (DPP) scheduler was proposed in [11], which aims to maintain the stability of the traffic queues, while ensuring the GFBR and an efficient resource utilization. To achieve this, a decision regarding the Physical Resource Block (PRB) allocation is made by DPP within a given slot time. The duration of this slot, t, and the amount of available resources, denoted as $\mathcal{A}$, depend on the configuration of the system, particularly the numerology and the bandwidth.

Let $Q_n(t)$ denote the backlog of the n^{th} queue at time slot t, and $\alpha_n(t)$ the scheduling decision, i.e. the number of PRBs assigned to the n^{th} flow. The traffic arrivals at the n^{th} queue are represented by $a_n(t)$ and $b_n(t)$, respectively. In general, the outgoing traffic can be defined as $b_n(t) = \hat{b}_n(\alpha(t), \omega(t))$, where $\omega(t)$ holds for a general random parameters, such as the Signal to Interference and Noise Ratio (SINR).

The average window time, defined by the 5QI, indicates the time over which the GFBR shall be calculated. The default value for Guaranteed Bit Rate (GBR) and Delay-Critical GBR resource types is set at 2 seconds for each standardized 5QI value. The scheduling decisions are made in much shorter time units, order of milliseconds, and the throughput requirement is thus defined as an average time objective in DPP. For

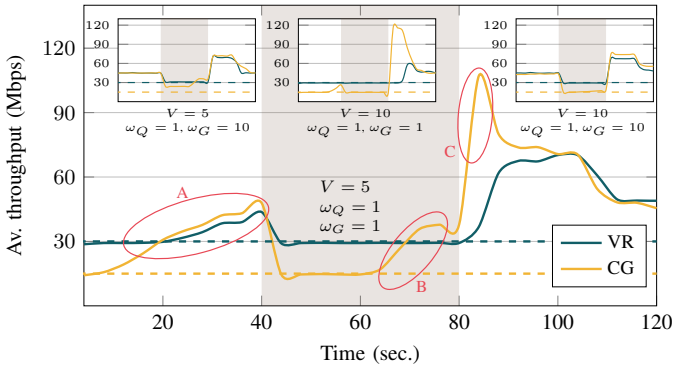

Fig. 1. Example of throughput performance upon different configurations.

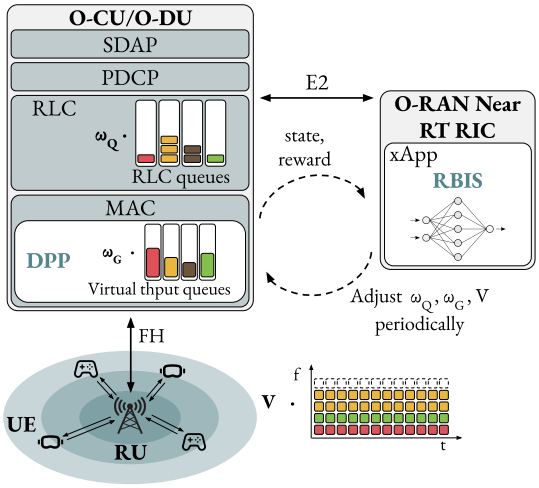

Fig. 2. Example of throughput performance upon different configurations.

this purpose, a virtual throughput queue, $G_n(t)$, is defined as follows: $G_n(t+1) = \max\{G_n(t) + D_n(t), 0\}$. It depends on the throughput deficit $D_n(t) = \Gamma_n - \rho_n(t)$, where $\rho_n(t)$ corresponds to the throughput obtained at the current slot, due to the scheduling decision and the SINR. As described in [11], the resulting problem can be tackled using the *min drift-plus-penalty* algorithm, which boils down to solving Problem 1 at every slot.

Problem 1:

$$\min_{\alpha(t)} \quad V \sum_{n\in\mathcal{N}} \alpha_n(t) + \omega_Q \sum_{n\in\mathcal{N}} Q_n(t)[a_n(t) - b_n(t)] \qquad (1)$$
$$+ \omega_G \sum_{n\in\mathcal{N}} G_n(t) D_n(t)$$

$$\textbf{s.t.} \quad \sum_{n\in\mathcal{N}} \alpha_n(t) \leq \Lambda \quad \forall t \qquad (2)$$

$$b_n(t) \leq Q_n(t) \quad \forall n \in \{1, \ldots, N\}, \forall t \qquad (3)$$

The first term of the objective function in Problem 1, the *penalty*, implies the minimization of the allocated resources. The second and third terms correspond to the so-called *drift*. The objective is to keep this *drift* as small as possible, thereby reducing the delay and ensuring that throughput requirements are met, while minimizing the resource usage via the *penalty* term. Furthermore, some configurable parameters are introduced. These are: V, which adjust the importance given to resource efficiency; and ω_Q and ω_G, which do the same for the goal of stabilizing RLC and virtual throughput queues, respectively.

The optimal value of these weights depends on the flow characteristics and the scenario. For instance, in the event of high traffic rates, the algorithm may prioritize the stabilization of RLC queues over GFBR compliance. Consequently, increasing ω_G in the objective function could be seen as a strategy to enhance the performance. On the other hand, if the system capacity is enough to meet the needs of all traffic flows, it may be a better policy to adjust the value of V according to the other weights. This allows for minimum resource usage, while ensuring GFBR and traffic queue stability.

Figure 1 illustrates the average throughput achieved by some UEs sending XR traffic. In particular, the figure shows the throughput observed by Virtual Reality (VR) and Cloud Gaming (CG) UEs for various illustrative and static configurations using the DPP scheduler. The specific configuration in question, i.e. a combination of V, ω_Q and ω_G, implies a distinct set of allocations and adjustments that influence resource distribution, throughput, and latency management. Therefore, every configuration parameter is designed to prioritize the optimization of a particular Key Performance Indicator (KPI). Dashed lines on Figure 1 correspond to throughput targets for each traffic type, i.e. their GFBR, and during the time illustrated by the shaded area (between seconds 40 and 80) channel conditions were worsened. Furthermore, red circles, A, B and C, highlight some regions where the behavior of the scheduler is most noticeably affected by the particular scheduler configuration. We argue that an appropriate reconfiguration in these regions might help to mitigate abrupt changes in the performance, caused by variations in the environment, such as channel degradation, or changes in user demands, in terms of either traffic volume or QoS requirements. As an example, increasing the ω_G parameter leads to a throughput flattening in region C, thereby enhancing the overall performance. Given the considerable number of potential configurations and the intricacy of their real-time scenario-specific adjustment, the deployment of a DRL agent to smartly configure the scheduling policy, enabling a more efficient selection process, would allow the system to autonomously adapt, and so achieve an optimal performance.

Figure 2 provides the overview of the proposed architecture. As discussed, the scheduler is responsible of the PRB allocation between UEs, considering the RLC queues and the GFBR, by maintaining the stability of virtual queues (scale of millisecond), with the additional goal of using resources efficiently. In this context, RBIS is deployed as an O-RAN *xApp*, enabling the reconfiguration of the scheduler in real time (scale of second).

IV. Reinforcement-Based Intelligent Scheduling

This section introduces RBIS, as an efficient solution to the problem depicted in Section III. First, we briefly introduce

the theoretical framework behind DRL and some of its basic concepts. We then discuss how it fits within the system model. Finally, we describe the specifics and technical details on RBIS' implementation of DRL.

The learning process in DRL can be described as a Markov Decision Process (MDP), a mathematical model that defines the interactions of a DRL system with a given environment and how it can be steered towards interactions deemed *better* [12]. An MDP can be structured as a decision-making algorithm, known as the *agent*, and an *environment* that the agent must interact with. The agent interacts with the environment by making decisions at discrete time steps: $\tau \in 0, 1, \ldots, \mathcal{T}$. The environment comprises a *state space* S, that defines every possible representation of its status (i.e., all possible values of all metrics deemed relevant to the environment), and an *action space* A, entailing all the possible decisions the agent can make.

At each time step τ, the agent is given, as input, a certain *state* $S(\tau) \in S$, i.e., a representation of the specific situation the environment is in. Based on this state, the agent takes an *action* $A(\tau) \in A$. This action entails making a decision, moving the environment to a new state $S(\tau+1) \in S$ and, as a result, receiving a *reward* $R(S(\tau), A(\tau)) \in \mathbb{R}$. This reward is the result of a *reward function*, which evaluates the fitness of the action for the given state, i.e., how good or bad the action is, considering the current state: $\mathcal{R} : S \times A \to \mathbb{R}$. The reward allows the agent to learn, steering its decisions to maximize the reward obtained over time.

Formally, the goal of the agent is to define a policy $\pi(A(\tau)|S(\tau))$ that describes the probability distribution of choosing the action $A(\tau)$ given the state $S(\tau)$, maximizing the total reward. Nonetheless, it is common to use the *discounted reward* function for this purpose, which uses a discount rate parameter ($\gamma \in [0, 1]$) to benefit agents that took good decisions in the past. Formally, the discounted return for a time step $G(\tau)$ is defined as $G(\tau) = \sum_{k=0}^{\mathcal{T}} (\gamma^k \mathcal{R}(S(\tau + k), A(\tau + k))$. Based on these definitions, DRL also leverages the *state-value function*, which defines the expected discounted return the agent gets by applying the policy π from state s onwards. Formally, the state-value function for state s and policy π is defined as $V(\pi, s) = \mathbb{E}_\pi[G(\tau)|S(\tau) = s]$. Finally, it is also necessary to define the *action-value function*, i.e., the expected discounted return from applying action a in state s, and following policy π onwards, or formally: $\mathbb{Q}(\pi, s, a) = \mathbb{E}_\pi[G(\tau)|S(\tau) = s, A(\tau) = a]$. The objective of DRL is to learn a policy π that is as close to the optimal one as possible.

To model the problem for DRL, we need to define how its elements and the underlying system model map into an MDP. The first element to model is the *observation space* O, which is a subspace of the state space that defines the parts of the environment the agent is able to observe. Formally, $O \subseteq S$, and hence, $O(\tau) \subseteq S(\tau) \forall t \in [0, \mathcal{T}]$. In this specific problem, the agent is allowed to observe the complete state space, and hence, $O = S$. The state comprises the number of RLC queues $N = |\mathcal{N}| \in \mathbb{N}$ that exist in the scenario, the MCS used for each of the flows M_n, the GFBR for each flow $\Gamma_n(t)$, the total number of available PRBs Λ, and the average buffer size for each flow B_n. S can thus be formally defined as:

$$S = \{N, M_1, ..., M_n, \Gamma_1, ..., \Gamma_n, \Lambda, B_1, ..., B_n\} \quad (4)$$

The objective of the DRL agent is to adjust the values of the three weights of the *min drift-plus-penalty* algorithm, namely, V, ω_Q, and ω_G. Hence, an action taken by the agent represents a set of values given to each of these three weights, respectively. Formally, the action space A is defined as:

$$A = \{V, \omega_Q, \omega_G\} \quad (5)$$

The final element to map from the problem domain to the MDP is the reward function $\mathcal{R}(S(\tau), A(\tau))$. For this particular problem, the DRL agent receives a reward based on the average throughput and resource efficiency achieved after the *min drift-plus-penalty* algorithm is executed with the weights previously chosen by the DRL agent. The specific reward function leveraged by the DRL agent combines the average value of the throughput across all flows with the fraction of free PRBs. Both of these rewards have been normalized between 0 and 0.5, leading to an overall reward between 0 and 1. Furthermore, to lead the agent towards desired states and ensure it does not take defective decisions (e.g., not assigning resources to maximize free PRBs), the resource efficiency part of the reward is not considered unless the GFBR is achieved. Formally, $\mathcal{R}(S(\tau), A(\tau))$ is defined as follows:

$$\mathcal{R}(S(\tau), A(\tau)) = \frac{1}{2} \cdot \frac{\sum_{n \in \mathcal{N}} \min\left(1, 1 - \frac{\Gamma_n - \bar{\rho}_n}{\Gamma_n}\right)}{N} + \\ + \frac{1}{2} \cdot \left(1 - \frac{\sum_{n \in \mathcal{N}} \bar{\alpha}_n}{\Lambda}\right) \cdot \mathbf{1}_{\{R_{thput}=1\}} \quad \forall n \in \mathcal{N} \quad (6)$$

$$\mathbf{1}_{\{R_{thput}=1\}} = \begin{cases} 1, & \text{if } \frac{\sum_{n \in \mathcal{N}} \min\left(1, 1 - \frac{\Gamma_n - \bar{\rho}_n}{\Gamma_n}\right)}{N} = 1, \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

Considering this MDP model, RBIS leverages the Proximal Policy Optimization (PPO) algorithm. In PPO, the DRL agent tries to directly approximate the optimal policy π by means of a neural network. More specifically, the DRL agent contains a neural network $\hat{\pi}$ with a set of weights θ, which takes a state as an input, and produces an action as an output: $\hat{\pi}(\theta, S(\tau)) = A(\tau)$. The training process leverages the reward to adjust θ so that the reward is maximized, hence approximating the optimal policy.

V. Evaluation

A. Evaluation setup

The performance of RBIS is assessed over a topology comprising one gNB and four UEs, all located in the same beam. To conduct the tests, we used 5G-LENA, which provides a detailed and realistic model of PHY and MAC layers of 5G NR. The configuration of the scenario is outlined in

TABLE I
CONFIGURATION OF THE EVALUATION SETUP.

Parameter	Value
Number of UEs per gNB	4
Carrier frequency	$4\,GHz$
Bandwidth	$80\,MHz$
gNB Tx power	$41\,dBm$
UE Tx power	$23\,dBm$
BS antenna height	$25\,m$ / $3\,m$
UE antenna height	$1.5\,m$ / $1\,m$
BS antenna array	1 TXRU: 4 × 8 (3GPP elements)
BS antenna element gain	$8\,dBi$
UE antenna array	1 TXRU: 1 × 1 (isotropic element)
UE antenna element gain	$0\,dBi$
BS noise figure	$5\,dB$
UE noise figure	$7\,dB$
Noise power spectral density	$-174\,dBm/Hz$
Duplexing mode	TDD
TDD pattern	DDDDU

TABLE II
CONFIGURATION OF THE XR TRAFFIC.

Virtual Reality traffic		Cloud Gaming traffic	
Number of flows	1	Number of flows	1
5QI value	87	5QI value	3
Average rate offered	$45\,Mbps$	Average rate offered	$30\,Mbps$
FPS	120	FPS	120
Guaranteed Bit Rate	$30\,Mbps$	Guaranteed Bit Rate	$15\,Mbps$

Table I. All UEs generate XR traffic, including VR and CG flows, whose characteristics are depicted in Table II. The DPP MAC scheduler was previously integrated within the simulator in [11]. The scheduling decisions provided by the scheduler are used to train RBIS, and afterwards RBIS' outcomes are validated over the simulation.

B. Evaluation results

Figure 3 compares the reward over time obtained with three methods: random tuning, where the values of V, ω_q, and ω_g are randomly selected, the trained RBIS DRL agent, and the RBIS DRL agent during training. This analysis serves three purposes: (1) to analyze RBIS' convergence time and its learning ability in prolonged time periods; (2) to assess whether the trained model obtains similar rewards to when it is in a training phase; and (3) to compare RBIS with a random tuning. Considering the convergence period, the results show

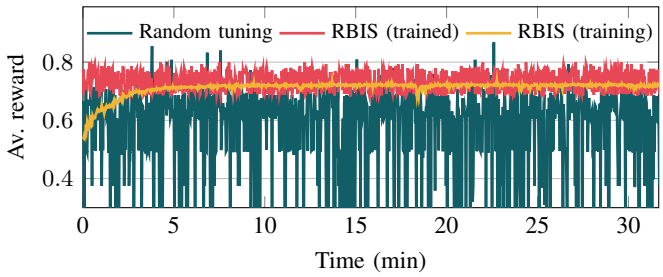

Fig. 3. Comparison of the reward obtained over time.

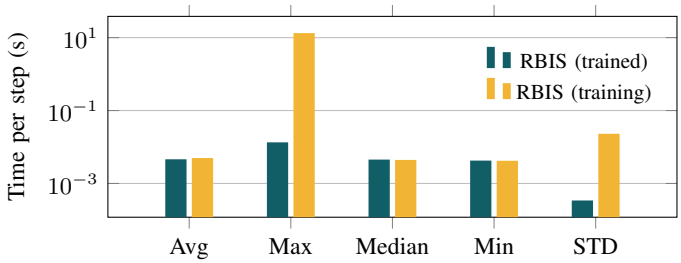

Fig. 4. Comparison of the time taken to take a decision (step).

that RBIS converges quickly, in 8 minutes and 20 seconds, and then continues with a slight but steady improvement, from the initial convergence reward of 0.719 to 0.731 at the end of the test. This is reflected by the trained agent, which achieves similar rewards to those obtained by the model at the end of training. The RBIS agent during training is not yet ready for decision-making, making early comparisons with other methods like reward assessment unfair. However, once fully trained, RBIS can be effectively compared to random-based tuning. The random-based agent achieves rewards between 0.305 and 0.731, with an average of 0.573 and a median of 0.580, and a standard deviation of 0.059. In contrast, RBIS achieves a narrower reward range of 0.702 to 0.774, with a higher maximum reward. RBIS has a median reward of 0.727, which is slightly lower than the random method's best reward, but an average reward of 0.733 surpasses the random method's highest values. Additionally, RBIS shows more stability with a standard deviation of 0.01.

We then compare the time required by RBIS to take a decision, i.e., for the DRL agent to take a step, as shown in Fig. 4. It is critical for the steps to be taken as quick as possible, as this allows RBIS to react to the status of the RAN and adjust the MAC scheduling accordingly within a sensible time interval. The analysis examines the decision-making times of RBIS during and after training. During training, RBIS takes an average of 4.976 ms to make a decision, with wide variability, ranging from 4.190 ms to 13,294.029 ms. Once trained, RBIS' decision times stabilize significantly, ranging from 4.241 ms to 13.419 ms with a median of 4.517 ms and an average of 4.602 ms. The trained model demonstrates more consistent behavior and consistently reacts quickly with an average decision time of 4.602 ms, indicating its efficient adaptation to changes. The results evince that, once trained, the response time of RBIS can be used within the time frame for GFBR calculation (2 seconds) to reconfigure the MAC scheduler.

Figures 5 and 6 show the throughput achieved and usage of resources, respectively. Both metrics are represented by boxplots, with the circles showing the average values. The figures compare the results obtained by the reconfiguration chosen by the trained agent against those from a random one. The results are classified according to the type of traffic, and every set of results includes 3 different channel states for the corresponding traffic: L (low), M (medium), and H

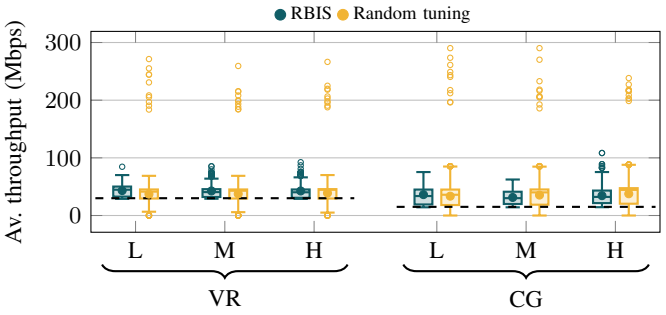

Fig. 5. Comparison of the average throughput achieved by each type of UE.

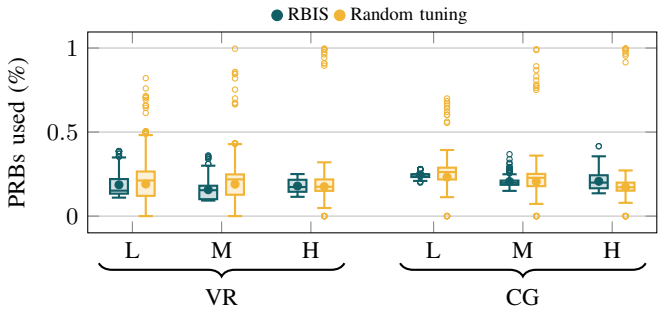

Fig. 6. Comparison of the PRB usage by each type of UE.

(high) quality. As can be seen, the throughput obtained by the trained agent remains close to the GFBR (dashed lines), as was expected. On the other hand, the random selection can yield extreme values, particularly when no resources are allocated or all resources are assigned to a single UE. Figure 6 shows that, while always ensuring the required GFBR, RBIS also reduces the usage of PRBs, as well as its variability. All in all, RBIS the results demonstrates that RBIS is able to tune the MAC scheduler according to traffic and channel variations, leading to more efficient usage of resources.

VI. Conclusion and Future Works

In this paper we have presented RBIS, the use of DRL for real-time 5G MAC scheduler reconfiguration. The proposed approach leverages the strengths of DRL to adaptively adjust the behavior of a MAC scheduler to optimize resource allocation between users with a wide variety of QoS requirements.

To assess the feasibility of our proposed solution, we have conducted experiments in a 5G cell populated with XR UEs, using a multi-objective Lyapunov-based MAC scheduler to manage available resources. The simulation results evince the robustness of our proposal, which consistently finds optimal configurations for the MAC scheduler. Furthermore, RBIS mitigates the risk of scheduler malfunction due to suboptimal configurations, which can arise from channel variability or the distinct user demands. In addition, we have also studied the overhead imposed by the DRL agent, and we have seen that it is kept at reasonable levels, thus ensuring the applicability of the scheduling policy.

In our upcoming work, we plan to extend the evaluation using more complex network scenarios. Furthermore, we intend to further explore enhancements in the reward function to balance additional network objectives and to fully integrate RBIS within O-RAN architecture as an *xApp*.

Acknowledgments

This work is supported by the European Commission's Horizon Europe, Smart Networks and Services Joint Undertaking, research and innovation program under grant agreement #101139282, 6G-SENSES project, as well as by the Spanish Government, with the projects PDC2022-133465-I00, PID2021-124054OB-C31, TED2021-130913B-I00 funded by MICIU/AEI/10.13039/50100011033 and the "European Union NextGenerationEU/PRTR", and 6GBLUR/JOINT (TSI-063000-2021-57). It has also received funds from the regional Cantabria and Extremadura Governments through the TCNIC program (2023/TCN/002) and Concepción Arenal program, grant UC-23-40, and the grant GR21133 from the Department of Economy, Science and Digital Agenda of the Government of Extremadura. It was also supported by the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"). CUP: J33C22 002880001.

References

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[2] O-RAN Alliance, "O-RAN: Towards an Open and Smart RAN," White Paper, 2018. [Online]. Available: https://mediastorage.o-ran.org/white-papers/O-RAN.White-Paper-2018-10.pdf

[3] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020.

[4] A. Mamane, M. Fattah, M. E. Ghazi, M. E. Bekkali, Y. Balboul, and S. Mazer, "Scheduling algorithms for 5g networks and beyond: Classification and survey," *IEEE Access*, vol. 10, pp. 51 643–51 661, 2022.

[5] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.

[6] I.-S. Comşa, S. Zhang, M. E. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea, "Towards 5g: A reinforcement learning-based scheduling solution for data traffic management," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1661–1675, 2018.

[7] R. Li, Z. Zhao, Q. Sun, C.-L. I, 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.

[8] S. Imtiaz, H. Ghauch, G. P. Koudouridis, and J. Gross, "Random forests resource allocation for 5g systems: Performance and robustness study," in *IEEE WCNCW 2018*, 2018, pp. 326–331.

[9] A. Chilmulwar and V. Sinha, "A novel machine learning based mac scheduler algorithm using arima," in *IEEE CCNC 2019*, 2019, pp. 1–8.

[10] X. Guo, Z. Li, P. Liu, R. Yan, Y. Han, X. Hei, and G. Zhong, "A novel user selection massive mimo scheduling algorithm via real time ddpg," in *IEEE GLOBECOM 2020*, 2020, pp. 1–6.

[11] N. Villegas, A. Larrañaga, L. Diez, K. Koutlia, S. Lagén, and R. Agüero, "Extending qos-aware scheduling in ns-3 5g-lena: A lyapunov based solution," in *Proceedings of the 2024 Workshop on Ns-3*, ser. WNS3 '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 54–59.

[12] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.