Flexible Reinforcement Learning Scheduler for 5G Networks

Aurora Paz-Pérez[†], Anxo Tato[†], J. Joaquín Escudero-Garzás[†], Felipe Gómez-Cuba[*].

[†] GRADIANT (Galician Research and Development Center for Advanced Telecommunications), Vigo, Galicia, Spain

[*]atlanTTic, Universidade de Vigo, Galicia, Spain

Email: {apperez, atato, jescudero}@gradiant.org, gomezcuba@gts.uvigo.es

***Abstract*—Resource management in 5G networks is becoming a highly complex task with the need to handle multiple devices and applications effectively with large data rates. In particular, optimizing scheduling algorithms improves resource allocation efficiency. To address this optimization, Artificial Intelligence (AI) techniques have recently shown their effectiveness compared to classical optimization algorithms. This work aims to design and implement a Reinforcement Learning (RL) MAC-layer scheduler that outperforms traditional methods in a potential real-world deployment. The main outcomes include a high-performing RL agent capable of adapting to diverse device types and handling a flexible number of user equipment (UE), which adds significant research value to the field since the current studies do not include a variable number of UEs.**

***Index Terms*—Artificial Intelligence, 5G, Reinforcement Learning, MAC scheduling, ns-3 simulator**

I. Introduction

A crucial factor in wireless networks is the efficient and fair distribution of resources to devices. This distribution is managed by a scheduler, which implements specific policies to prioritize certain users over others. This approach enhances both real-time efficiency and adaptability. Leaving aside the spatial component of MIMO (Multiple Input, Multiple Output), the physical layer resources are organized within a two-dimensional grid of time and frequency that the scheduler has to optimally distribute among the users. It is difficult to manage 5G networks with traditional scheduling algorithms due to the large variety of devices, which increases the network complexity. To address this problem, integrating Machine Learning (ML), especially Reinforcement Learning (RL), has emerged as a promising solution for making decisions in the Medium Access Control (MAC) scheduling [1], [2].

A. Review of existing work

Among the ML paradigms for scheduling, RL algorithms predominate due to i) the RL agent's capacity for interacting with the environment through actions; ii) its ability to learn through trial and error, adjusting its decision-making strategy and refining its policy over time, and iii) its ability to find an optimal policy that maximize long-term rewards. In the literature, we find that Double Deep Q Network (DQN) [3] and Deep Deterministic Policy Gradient (DDPG) [4]–[6] stand out from the different types of 5G MAC scheduling RL algorithms. In [4], the scheduler proposed for New Radio (NR) gNB considers two types of numerologies and different Quality of Service (QoS) according to the type of traffic. In [5] the scheduling achieves a long-term QoS trade-off between different types of traffic. The authors of [6] present a DDPG scheduler that uses expert knowledge (K-DDPG) to reduce the convergence time, an aspect necessary to work in a real system. This paper includes experimental results, though the setting consists of one LTE eNB and two UEs. Another popular approach is Actor-Critic (AC) learning [7]–[9], where the actor is responsible for the action selection and the critic predicts anticipated returns based on the policy. While Q-Learning requires that all actions must be tested in all states, in AC systems exploration is fully determined by the action probabilities of the actor. In [7], the authors develop an AC agent using a pointer network architecture to provide flexibility and scalability and it is implemented within the Nokia Wireless Suite simulator [10]. The authors of [8] propose an AC-based scheduling framework for 5G networks to select a resource allocation rule. The work of [9] presents a distributed AC multi-agent scheme whose agents collaborate to perform optimal scheduling, allowing the scheduler to be deployed in a resource-limited network. Also, a recent study [11] addresses scheduling in a multi-hop millimeter-wave mesh. They use a model-free algorithm called Adaptive Activator RL, which determines the subset of links to be activated during each time slot and the power level for each link.

B. Contributions

A limitation observed in many RL-based schedulers is their inflexibility, because their design is tailored to a specific number of User Equipment devices (UEs). This approach dictates the dimensions of inputs and outputs in the underlying neural network. One distinctive aspect of this work lies in a better adaptation to environmental changes compared to classical scheduling algorithms, such as Proportional Fairness (PF), since more features are taken into account. But the most significant breakthrough is the formulation of a versatile framework capable of managing a variable number of UEs in an RL-based scheduler. Considering all these aspects, the main objectives of our work are listed below:

- Design and implement an RL agent that performs the functions of a MAC layer scheduler in a 5G base station.
- Create a standardized simulation environment capable of realistic radio resource allocation.

- Compare the RL agent with conventional baseline algorithms, aiming for enhanced efficiency and adaptability.
- Elevate the current state-of-the-art by introducing a flexible RL agent capable of managing a varying number of UEs.

The article is structured as follows. After this introduction, Section II presents the proposed RL-based scheduling system as well as the simulation environment. Then, Section III evaluates the proposed scheme in two different scenarios. Lastly, the main conclusions and future work are drawn in Section IV.

II. RL-BASED SCHEDULING SYSTEM

We developed the RL-based scheduling system showed in Fig. 1, consisting of a cellular simulation environment, an RL agent, a learning mechanism, a set of features and a reward function (RF). The characteristics of each of these components are described below:

A. Environment

We consider a 5G cellular network simulated with the 5G-LENA module of the ns-3 simulator. This module is aligned with NR Release 15 TS 38.300 and allows us to emulate the communications of a base station (gNB) with a set of N UEs. Particularly, 5G uses a Orthogonal Frequency Division Multiple Access (OFDMA) physical layer where the MAC scheduler assigns Physical Resource Blocks (PRBs) of 12 subcarriers to the instantaneous number of active UEs (N_a), as depicted in Fig. 1.

The simulated scenarios consist of hexagonal 5G cells as depicted in Fig. 2. The RL scheduler controls a urban microcell (UMi) with the gNodeB (gNB) in its center at height h_{gNB} m and a maximum number of N UEs at height h_{UE}, in random positions and moving on average v_{UE} m/s. Both UEs and gNB have isotropic antennas. We conducted simulations where UEs have one of the following profiles:

- Voice/video application: rate $\geq r_{vvmin} = 7.5$ MB/s, Packet Delay Budget (PDB_{vvmax}) = 100 ms.
- Low latency application: rate $\leq r_{llmax} = 100$ kB/s, PDB_{llmax} = 30 ms.

B. RL Agent

An RL agent is a type of Markov decision scheme that, for a given environment state S_t, chooses an action A_t. The action on the environment produces the next state S_{t+1}, and a value of the reward function R_t as shown in Fig. 3. A good policy $\pi(A_t|S_t)$ chooses a sequence of actions given the current states in order to maximize the long-term sum of rewards R_t. Therefore, in our environment, the RL agent faces the following optimization problem:

$$\max_{\pi, A_t} \quad R_t(A_t, S_t) \tag{1a}$$

$$\text{s.t.} \quad r_{vv} \geq r_{vvmin} \tag{1b}$$

$$PDB_{vv} \leq PDB_{vvmax} \tag{1c}$$

$$PDB_{ll} \leq PDB_{llmax}, \tag{1d}$$

where the objective is to maximize the R_t designed in the following sections by selecting the optimal actions A_t, subject to the constraints for each application ("vv" represents voice/video and "ll" low latency).

In RL, the state-space of the environment is enormous or unknown for conventional optimization, so ML is applied. For the RL agent, we use the Proximal Policy Optimzation (PPO) algorithm [13] from the open-source library Ray RLlib. Ray RLlib gives a rich set of RL algorithms, distributed training capabilities and an interface for managing RL experiments. The PPO is an AC, on-policy and model-free algorithm with many benefits, as simplicity, stability, sample efficiency and scalability. This algorithm is composed of two neural networks, the actor and the critic, represented in Fig. 4. The actor network learns the policy $\pi_\phi(A_t|S_t)$. The critic network "learns to train" the first network, by predicting the expected return obtained by acting with the current policy starting from a specific state. In this type of on-policy algorithms, the current policy is first executed during a number of interactions and these experiences are stored in a rollout buffer. Then this data is used to train the two neural networks (the actor and the critic) with a gradient descent algorithm.

C. Flexible Learning Mechanism

Neural networks' inputs and outputs must be vectors of a fixed length. Most research papers propose schedulers that only manage a fixed number of UEs [14]. To avoid this shortcoming, we propose a Flexible Learning Mechanism (FLM) to support a variable number of UEs, both in training and operating time, having a fixed number of inputs and outputs of the PPO neural networks.

In our scenario, the input features received by the neural network (represented as sub-observation in the Fig. 1) are composed of three metric types: individual variables of a fixed number of W UEs, global statistics of all UEs and global cell features. Then, the neural network is fed with a batch of $K = \lceil \frac{N_a}{W} \rceil$ sub-observations. The output of the neural network is a batch of K actions. They are aggregated (to obtain N_a values) and normalized with the SoftMax function to obtain a valid action to be sent to the environment (list of percentages of PRBs assigned to each active UE).

One drawback of this solution is that some information is lost since the neural network never sees the complete landscape with the individual metrics of all UEs. However, the possibility of choosing the value of W brings flexibility to the system, allowing the RL agent to manage a dynamic number of UEs. Selecting a small W allows us to have simpler neural networks (less inputs and outputs), reducing the training and inference time. Moreover, W can be much smaller than N (our experiments show that the RL-based scheduler can work even with a value of just $W = 1$). In addition, the batch processing is a natural way of leveraging the parallel processing capabilities of Graphical Processing Units (GPUs). With the proposed FLM, a variable number of UEs, N, can be handled by just increasing or decreasing the size of the batch K. It should be noted that sub-observation size W is a design

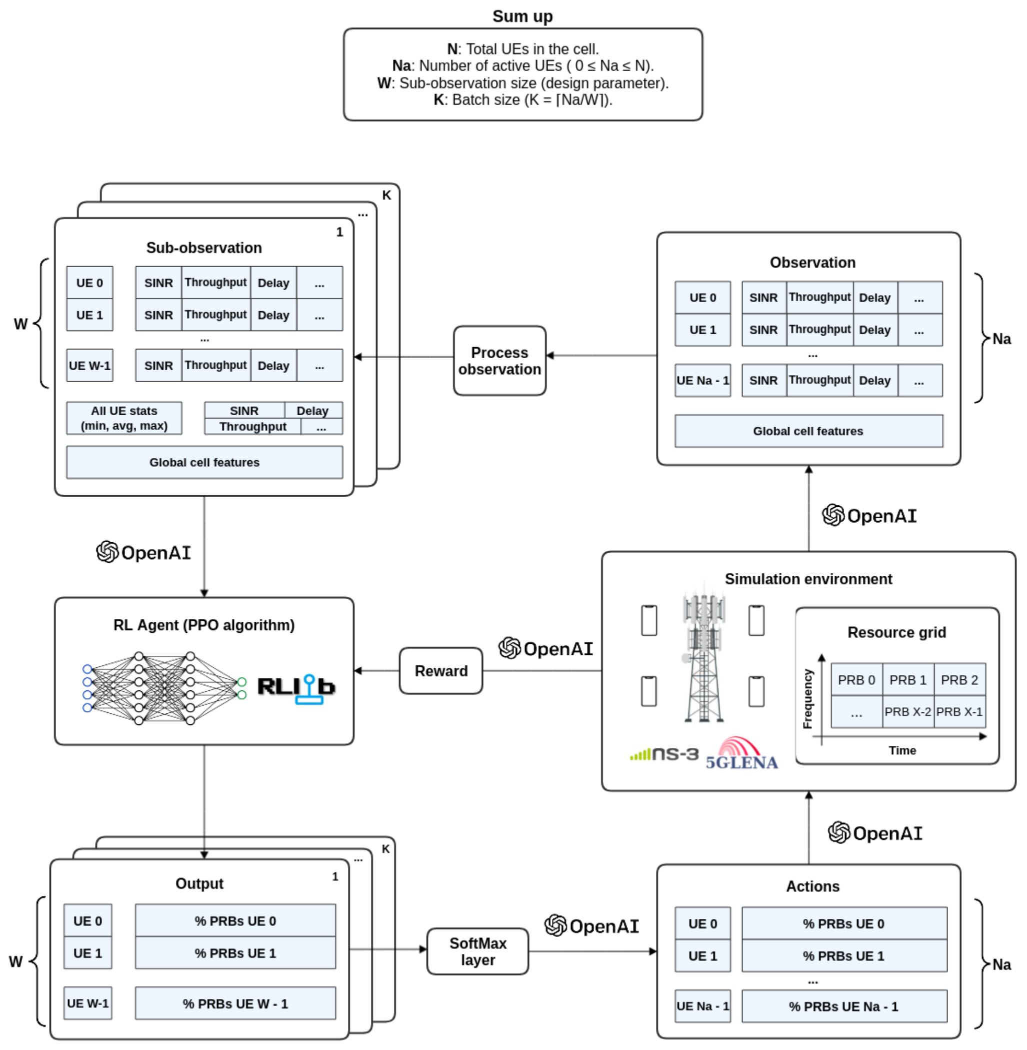

Fig. 1. Architecture of the RL-based scheduling system.

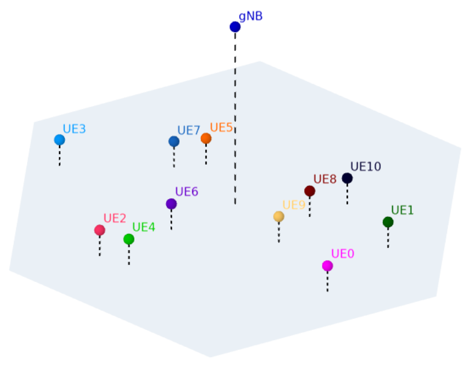

Fig. 2. Simulated scenario with a gNB and 11 UEs.

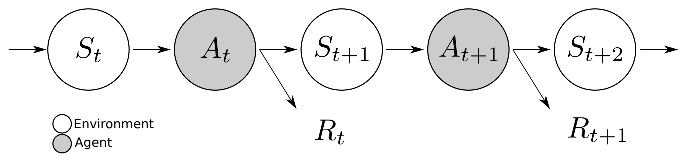

Fig. 3. RL interactions with the environment.

parameter that must be chosen and fixed in advance, since it conditions the size of the neural networks and, its modification would require a new re-training of the RL agent.

In summary, the process (represented in Fig. 1) consists of the following steps (blocks in **bold**):

- The sub-observation size W is provided to the **RL Agent** so it scales the number of inputs and outputs of the neural networks accordingly.
- In each step UE-specific metrics and global cell features are obtained from the **Simulation environment**, represented by the **Observation** block.

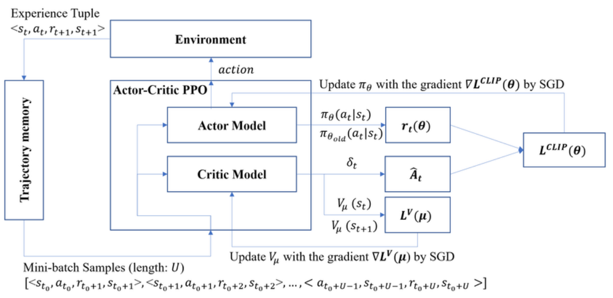
Experience Tuple
$<s_t, a_t, r_{t+1}, s_{t+1}>$
Environment
action
Actor-Critic PPO
Update π_θ with the gradient $\nabla L^{CLIP}(\theta)$ by SGD
Trajectory memory
Actor Model
Critic Model
$\pi_\theta(a_t|s_t)$
$\pi_{\theta_{old}}(a_t|s_t)$
$r_t(\theta)$
$L^{CLIP}(\theta)$
δ_t
$\hat{A}_t$
$V_\mu(s_t)$
$V_\mu(s_{t+1})$
$L^V(\mu)$
Update V_μ with the gradient $\nabla L^V(\mu)$ by SGD
Mini-batch Samples (length: U)
$[<s_{t_0}, a_{t_0}, r_{t_0+1}, s_{t_0+1}>, <s_{t_0+1}, a_{t_0+1}, r_{t_0+2}, s_{t_0+2}>, \dots, < a_{t_0+U-1}, s_{t_0+U-1}, r_{t_0+U}, s_{t_0+U} >]$

Fig. 4. Scheme of the PPO algorithm (taken from [12]).

- The interface between the **Simulation environment** and the **RL Agent** creates a batch of **Sub-observations**, which contains the three types of features described previously. This process is represented by the **Process observation** block.
- All the sub-observations are introduced to the **RL Agent** in a batch of size K.
- The **Output** of the **RL Agent** (batch of size K) is aggregated and normalized with a **SoftMax layer**, resulting in the **Actions**. Then the **Actions** (list of percentages of PRBs assigned to each UE) are provided to the **Simulation environment** as the next scheduling decision.

D. Features

The created simulation environment exposes features to the agent (observation). For each time instant, it obtains individual metrics of each UE (queues state, throughput, delay, SINR, etc) and global cell features (applications information, number of UEs (N), bandwidth, etc). With this information we compute the statistics of the global UE population (minimum, average and maximum of each metric). Moreover we calculate other global information, such as Jain's Fairness index (JFI) [15]. The complete list of dynamic features is shown in Table I. This list is updated at each time instant with the new values collected from the environment.

TABLE I
INPUT FEATURES TO THE RL AGENT.

Features per UE and min/avg/max stats	***Global cell features***
Queues: Number of bytes in buffers, traffic pending flag, packets transmitted, packets received and packets lost	**Applications:** Type flag, avg packets, PDB and number of active UEs in each application type
Channel: Throughput, delay, SINR	**Environment:** N, W, bandwidth, transmit power, TDD/FDD flag, TDD pattern, DL enabled, UL enabled, notched RBs
Min, avg and max values of the previous metrics	JFI index

E. Reward function

Other objective of this work is to design the most suitable reward function $R_t(A_t, S_t)$. This function provides a quantitative metric indicating the desirability of each behavior. By choosing the algorithm, the features and the R_t, our agent is fully defined. We followed an iterative reward-design process, starting with a basic proposal and modifying it based on experimental observation. We begin with the following general structure

$$R_t(A_t, S_t) = \begin{cases} -1 & \text{UE with empty buffer but other UEs not.} \\ 0 & \text{all buffers are empty.} \\ V^{\text{packets}} & \text{UE buffer has packets.} \end{cases} \tag{2}$$

where the first case penalizes the allocation of resources to inactive UEs when active users exist. The second case is a neutral reward as there are no UEs with traffic. For the rest of our work, we focused on designing different formulas for the reward value of UEs with packets $V^{packets}$.

Next, we sketch the main steps towards the final expression of $V^{packets}$. We start with the following expression:

$$V^{\text{packets}}_{\text{CQI and HoL}} = \frac{CQI[i]+1}{16} - \frac{1}{2*N_a} * \sum_{k=0}^{N_a-1} \frac{HoL[k]}{PDB[k]}. \tag{3}$$

This expression depends on the values of the Head of Line (HoL) delay and the channel quality indicator (CQI) of the selected UE i, $CQI[i]$. It seeks to prioritize UEs with better signal quality without sacrificing the average delay normalized by UEs' PDB. We observe that this reward, in case of network congestion, discriminates certain UEs starving them of resources. To solve this problem, we multiply (3) by the term $\log_2(\frac{PDB}{HoL})$. In this way, if the selected UE has a very small delay compared to the average, the obtained reward worsens. On the contrary, if the selected UE has a delay greater than average, a higher reward should be provided so the UE is preferentially selected.

During our experiments, these functions did not provide satisfying results. Therefore we replace the logarithmic function by an exponential term. This is due to the fact that, with the logarithmic function, the reward grows with a very small slope in case of congestion. That does not guarantee choosing the UE with highest delay, thus causing more congestion. However, by simply substituting the logarithmic function for the exponential term, it occurs that, in case of high average delays, the agent forces to defer an UE to get a better reward a posteriori. To deal with this problem, we looked for the R_t to reduce the average normalized delay, in this way not discriminating any UE.

In addition, another relevant parameter to take into account is the percentage of occupied buffer ($\%buffer$). Minimizing this parameter helps to reduce packet losses. With all these considerations, the final version of R_t was designed combining the following expression with (2) where $i = UE_{selected}$:

$$V_{\text{final}}^{\text{packets}} = a - b - c + 2 \text{ where } \begin{cases} a & = 2^{\frac{CQI[i] * HoL[i]}{PDB[i]}}, \\ b & = 2^{\frac{CQI[i]}{N_a} * \sum_{k=0}^{N_a - 1} \frac{HoL[k]}{PDB[k]}}, \\ c & = 2^{\frac{max(\%\texttt{buffer})+1}{\%\texttt{buffer}[i]+1.01} * 1.01}. \end{cases} \tag{4}$$

Our experiments (see Section III) show that (4) is the reward function that provides the best overall results in terms of throughput, delay, fairness and packet loss.

F. Integration

The complete system is depicted in Fig. 1. It shows the interactions between the elements of the architecture. Since 5G-LENA is not compatible with OpenAI Gym, an extra module called ns3-gym [16] was adapted to our use case. This module requires to define different elements: observation space, action space, end condition of the simulation, function to create the observations, reward function and the function that executes the actions.

The integration of this module with RLlib requires some modifications. The PPO implementation in RLlib allows to receive a batch of sub-observations, provided by Gym, as an input to the agent. As the agent receives an input batch of size K, it returns the actions also in batch of size K. Then we have to group these actions to send them to the environment jointly. In this way, the environment is not aware of the existence of the FLM, so it can communicate with all types of es of RL agents. To summarize, the environment sends one observation and one reward at each time instant and receives the actions grouped in one item while the agent operates with batches.

III. Evaluation

We consider a single cell with a different number of N UEs with voice/video and low latency applications with $h_{\text{gNB}} = 10$ m, $h_{\text{UE}} = 1.5$ m and $v_{\text{UE}} = 5$ m/s. The numerology is 0 (subcarrier spacing of 15 kHz). The gNB employs the n70 band in FDD mode with bandwidth of 10 MHz for both downlink and uplink. The simulations run over 2 s with a step of 1 ms between re-assignations of the PRBs. It should be noted that a hyperparameter tuning is performed. Specifically, the hyperparameters are the Generalized Advantage Estimate (GAE) parameter, the surrogate slipping parameter, the learning rate, the number of Stochastic Gradient Descent (SGD) iterations in each loop, the total SGD batch size and the training batch size.

Firstly, we compare our RL-based scheduler with other classical schedulers [17] already implemented in 5G-LENA, i.e. PF, Round Robin and Maximum Rate. It was found that PF is the classical algorithm that performs best in our environment. For this reason our proposed RL-based schedulers will be compared with the PF algorithm in this evaluation.

Secondly, one important aspect that should be taken into account is the training and inference time. The FLM allows a reduced size in the RL agent's neural networks, independently of the maximum number of users N. The sub-observation size, W, determines the number of inputs and outputs of the neural networks. It is a design parameter of the RL-based scheduler that cannot be changed without a model re-training. Moreover, the processing of the information is done in parallel, in batches of variable size K. These aspects allow a reduction in training and inference time compared with rigid RL schedulers tuned for a big number of UEs.

To show the improvement in training time, a experiment has been made. The RL-agent is trained in two scenarios, with $N = 8$ and $N = 4$ UEs. For each scenario, three different sub-observation sizes W are tested: 4, 2, and 1. The obtained training times are shown in Table II. It can be seen that the time decreases when more sub-observations are included (for smaller W) since they are processed in batch. Because the neural networks size is linear with W, a similar reduction in the inference time would be obtained.

The performance evaluation of the proposed RL-based scheduler is done by means of two different scenarios. A simpler one, with $N = 4$ UEs and a more complex one with up to $N = 11$ UEs. Each one has different application types, as shown in Tables III and V. Regarding the sub-observation size, W, values of 1, 2 and 4 are tested in each scenario. The number of instantaneous active users at each time step N_a and thus K are automatically determined.

The selection of the first scenario has two objectives. On the one hand, it shows that an RL-based scheduler with the PPO algorithm can outperform the well-known classical PF algorithm. On the other hand, it shows the good results obtained using our FLM, even if the agent only receives specific information from a subset of W UEs in each item of the batch (note that global statistics of all the active UEs N_a are also provided).

Regarding the second evaluation scenario, a higher number of UEs is selected, $N = 11$, to obtain a best approximation to a real scenario. We have not chosen a much larger value since the simulation time would be extremely long.

A. Scenario with 4 UEs

We first analyze the scenario with $N = 4$ UEs, whose arrangement in space is similar to Fig. 2. Moreover, the types of applications, their rate and when they start and finish transmitting are configured as illustrated in Table III. No UE leaves the cell during the simulation and, according to Table III, the range of active UEs is from 0 to 3. This scenario allows to see the difference between choosing different W values. The

TABLE II
Time improvement using the Flexible Learning Mechanism.

Number of UEs N	*Sub-observation size W*	*Training time (s)*
8	4	118 840
	2	11 712 (90% faster)
	1	5 844 (95% faster)
4	4	15 580
	2	5 712 (63% faster)
	1	2 929.7 (81% faster)

TABLE III
APPLICATION DIAGRAM FOR 4 UEs.

UE	*Start time*	*Finish time*	*Type*	*Rate*
0	0.10 s	1.70 s	Voice/video	12.5 MB/s
1	0.25 s	2.00 s	Voice/video	15.1 MB/s
2	0.40 s	0.60 s	Low latency	60 kB/s
3	1.50 s	1.75 s	Low latency	85 kB/s

agents are going to manage 4 UEs in sub-observations with size $W = [4, 2, 1]$ UEs. In the first case, all the information is available as inputs of the agent. The other two are examples of the FLM, having the first one more specific information than the last one.

The metrics obtained are illustrated in Table IV. In this scenario, using the PPO algorithm the delay drops in some cases to almost half compared to the PF algorithm. The throughput increases in all cases by more than 10%. This scenario verifies the advantages of using an RL-based MAC scheduler. It adapts to the typology of the different applications, achieving a great improvement in terms of throughput and delay compared with the PF classical algorithm. Moreover, the scenario demonstrates the good performance of the proposed FLM. Even if only one UE per sub-observation is given to the agent ($W = 1$), it still learns a good policy, as can be concluded from the results of Table IV. The search for an optimum value W is left as future work.

B. Scenario with 11 UEs

Finally, a scenario with $N = 11$ UEs is studied. The distribution of the devices in the space is illustrated in Fig. 2 and the application diagram is reflected in Table V. Similarly, no UE leaves the cell during the simulation and, according to Table V, the range of active UEs is from 0 to 7.

In this case we analyze the fact that the total number of UEs N is not a multiple of the number of UEs per sub-observation W. Note that in this case we approach a more realistic scenario taking into account a greater amount of UEs. The ratio of information per sub-observation versus the total is smaller in this case since the number of UEs is larger and W takes the same values, compared to the previous scenario.

Through the results we can observe how the RL agent maintains lower buffer occupancy compared to the PF algorithm, preventing network congestion and packet loss. It can also be seen how the delay is reduced compared to the PF algorithm, especially for low latency applications. This improvement can be seen in Fig. 5 with the maximum delays over time. The

TABLE IV
PERFORMANCE FOR SCENARIO WITH 4 UEs.

Scheduler	*Average throughput*	*Average delay*
Classical: PF	8.69 Mbps	13.4 ms
RL: PPO with $W = 4$	9.83 Mbps	7.64 ms
RL: PPO with $W = 2$	10.4 Mbps	8.98 ms
RL: PPO with $W = 1$	10.0 Mbps	7.64 ms

TABLE V
APPLICATION DIAGRAM FOR 11 UEs.

UE	*Start time*	*Finish time*	*Type*	*Rate*
0	0.10 s	1.70 s	Voice/video	12.5 MB/s
1	0.40 s	1.80 s	Voice/video	18.8 MB/s
2	0.20 s	2.00 s	Voice/video	15.1 MB/s
3	0.25 s	1.80 s	Voice/video	11.3 MB/s
4	0.30 s	1.90 s	Voice/video	12.5 MB/s
5	0.40 s	0.60 s	Low latency	80 kB/s
6	1.50 s	1.75 s	Low latency	70 kB/s
7	0.50 s	0.80 s	Low latency	60 kB/s
8	0.80 s	1.00 s	Low latency	85 kB/s
9	1.25 s	1.65 s	Low latency	75 kB/s
10	1.70 s	1.90 s	Low latency	65 kB/s

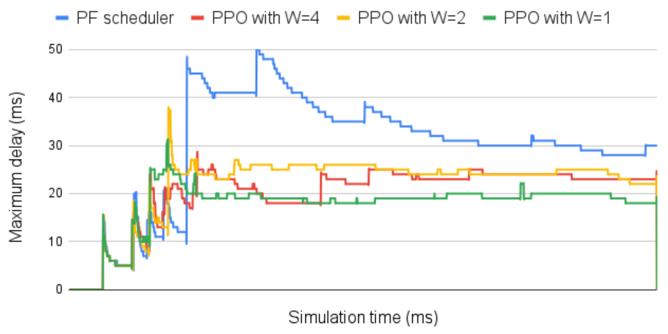

Fig. 5. Maximum delay (ms) for 11 UEs scenario.

RL agents consistently achieve maximum values near 30 ms, whereas the PF algorithm peaks at around 50 ms. Hence, despite not achieving substantial average differences, the improvement provided by the RL agents is deemed significant for reducing network congestion. Furthermore, Fig. 6 illustrates that the PF algorithm experiences high buffer occupancies, whereas the RL agents consistently maintain low and stable levels. This also verifies the enhancement in mitigating buffer congestion. Despite handling a much larger number of UEs, these RL-based schedulers either outperform or match the performance of the well-known classical PF algorithm.

In summary, through these experiments it can be seen how RL-based schedulers provide a better performance than classical schedulers, such as PF. It was possible to validate that the RL agent, in which $W = 2$, achieves optimal metrics while preserving the simplicity of the network. This is a desirable aspect to reduce computation times given that it is a process that must work in real time. Moreover, the simulations show that the proposed FLM enables to build simpler RL-based schedulers without loss of performance. With FLM the RL agents can manage a variable number of UEs with neural networks of bounded complexity.

IV. CONCLUSIONS AND FUTURE WORK

The MAC-layer scheduler is a breaking point in a cellular network, whose optimization is of paramount importance. Using RL-based schedulers for this task provides certain benefits such as real-time adaptability to the state of the network. It

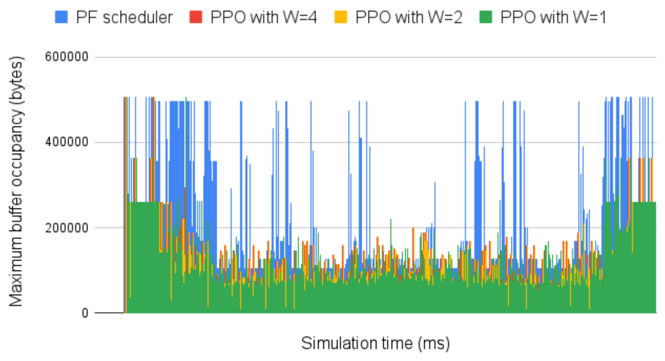

Fig. 6. Maximum buffer occupancy (bytes) for 11 UEs scenario.

also provides the possibility of balancing a wide range of network metrics at the same time through the RL agent's neural networks.

The main work of this paper is on the design and development of an RL-based scheduler that allocates radio resources. It improves the performance of traditional algorithms, such as PF, in terms of metrics like throughput or delay. It also adapts well to the different types of applications that arise in a realistic environment. The simulation results using the 5G-LENA module verify the effectiveness of the proposed scheduler, embracing the designed reward function, the selected features and the PPO RL algorithm.

On the other hand, the current state of the art is advanced by designing and implementing a Flexible Learning Mechanism (FLM) that allows an RL-based scheduler to handle a flexible number of UEs. This completely functional scheduler comes also with a remarkable improvement in training and inferring time, enabling the adoption of this RL-based approach for MAC-layer schedulers ready for being integrated in real cellular networks.

A future work that could be performed is a detailed analysis of the optimal values of the variable W for different N as well as the configuration of the N applications. Due to the long training time required to perform this analysis, no further tests were performed to determine the optimal values within this work. Another future direction of this research involves implementing the developed RL-based scheduler on a real base station. The open-source NR library OpenAirInterface integrates the NVIDIA Aerial as a hardware accelerator. This element would allow the scheduler to run on GPU, decreasing the inference times of the neural network. In this way, tests could be performed with a larger number of connected UEs using a real base station.

Acknowledgment

This research was supported in part by the Spanish Center for the Development of Industrial Technology (CDTI) and the Spanish Ministry of Economy, Industry and Competitiveness under grant CER-20191015 Open-VERSO, by the Galician Agency of Innovation (GAIN) under grant IN854A2020/01 Facendo 4.0 and the Spanish Ministerio de Ciencia e Innovación (MICINN) PID2021-122483OA-I00.

References

[1] Y. Arjoune and S. Faruque, "Artificial Intelligence for 5G Wireless Systems: Opportunities, Challenges, and Future Research Direction," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, pp. 1023–1028.

[2] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial Intelligence-Enabled Cellular Networks: A Critical Path to Beyond-5G and 6G," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020.

[3] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to Schedule (LEASCH): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5G MAC Layer," *IEEE Access*, vol. 8, pp. 108 088–108 101, 2020.

[4] S.-C. Tseng, Z.-W. Liu, Y.-C. Chou, and C.-W. Huang, "Radio Resource Scheduling for 5G NR via Deep Deterministic Policy Gradient," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[5] J. Li and X. Zhang, "Deep Reinforcement Learning-Based Joint Scheduling of eMBB and URLLC in 5G Networks," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1543–1546, 2020.

[6] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, "Knowledge-Assisted Deep Reinforcement Learning in 5G Scheduler Design: From Theoretical Framework to Implementation," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2014–2028, 2021.

[7] F. AL-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio Resource Scheduling with Deep Pointer Networks and Reinforcement Learning," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.

[8] I.-S. Comşa, R. Trestian, G.-M. Muntean, and G. Ghinea, "5MART: A 5G SMART Scheduling Framework for Optimizing QoS Through Reinforcement Learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1110–1124, 2020.

[9] D. Corcoran, P. Kreuger, and M. Boman, "A Sample Efficient Multi-Agent Approach to Continuous Reinforcement Learning," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 338–344.

[10] Nokia, "nokia/wireless-suite," 2020, accessed September 14th, 2023. [Online]. Available: https://github.com/nokia/wireless-suite

[11] B. Gahtan, R. Cohen, A. M. Bronstein, and G. Kedar, "Using Deep Reinforcement Learning for mmWave Real-Time Scheduling," 2023.

[12] H.-K. Lim, J.-B. Kim, J.-S. Heo, and Y.-H. Han, "Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices," *Sensors*, vol. 20, no. 5, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/5/1359

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

[14] S. Mollahasani, M. Erol-Kantarci, M. Hirab, H. Dehghan, and R. Wilson, "Actor-Critic Learning Based QoS-Aware Scheduler for Reconfigurable Wireless Networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 45–54, 2022.

[15] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," 1998.

[16] P. Gawłowicz and A. Zubow, "Ns-3 Meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 113–120. [Online]. Available: https://doi.org/10.1145/3345768.3355908

[17] B. G. Lee, D. Park, and H. Seo, *Wireless Communications Resource Management*. Wiley, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1002/9780470823583