

Reinforcement Learning for Radio Resource Management in O-RAN

Caio P. Galdino*, Rodrigo S. Couto*, Dianne S. V. de Medeiros†, Igor M. Moraes†, and Diogo M. F. Mattos†

*Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - Rio de Janeiro, RJ, Brazil

†Universidade Federal Fluminense - PPGEET/PGC/MídiaCom - Niterói, RJ, Brazil

Email: {galdino, rodrigo}@gta.uff.br, {diannescherly, igormoraes, diogo.mattos}@id.uff.br

Abstract—Open RAN is a groundbreaking approach that trends to revolutionize mobile network communication. Open RAN steers in a new era of data-driven decision-making RAN management by introducing intelligent controllers that adapt the Radio Access Network (RAN) in near-real time. As a consequence of the current surge in network traffic, the role of Artificial Intelligence in optimizing the RAN becomes paramount. The fifth-generation mobile networks (5G) further enhance this potential by enabling the allocation of clients through network slices for services such as eMBB, URLLC, and mMTC services. This paper presents a radio resource allocation policy that leverages Reinforcement Learning to make intelligent decisions that align with the objectives of each service in the network. The proposed innovative approach reduces user allocation waiting time by 20% compared to the traditional Round-Robin (RR) policy.

Index Terms—Open RAN, Reinforcement Learning, Radio Resource Allocation, 5G Network Slicing

I. INTRODUCTION

Radio Access Networks (RANs) are used to interconnect mobile devices with the core network that connects them to the Internet. In this context, Open RAN initiatives, such as the O-RAN architecture, emerge as an innovation, standing out for the desegregation, virtualization, and standardization of interfaces, allowing the application of Artificial Intelligence (AI) solutions in various stages and RAN protocols [1]. The O-RAN architecture defines RAN Intelligent Controllers (RICs), enabling the orchestration of simultaneous services and efficient and dynamic management of resources in network slices. RICs are divided into two main categories: Near Real-Time RIC (Near-RT RIC) and Non Real-Time RIC (Non-RT RIC). The former operates in time intervals greater than 10 ms and less than 1 s, while the latter operates in time intervals greater than 1 s. These controllers have the ability to dynamically adjust network parameters to meet service objectives, adapting to different scenarios. For example, it is possible to adjust parameters to meet the requirements of different network slices [1]. This work focuses specifically on the slices defined by the International Telecommunication Union

(ITU) for 5G: Enhanced Mobile BroadBand (eMBB), Ultra-Reliable Low Latency (URLLC), and massive Machine Type Communication (mMTC) [2]. These slices provide efficiency in delivering services to users with different requirements [1].

Allocation policies are traditionally complex, long, and challenging to implement, subject to human manipulation errors due to low-level programming of network objectives. This difficulty motivates the search for a more efficient, adaptive, and reliable approach, as advancements in network technology and the high availability of data enable these changes. The work proposed by Bonati et al. [3] develops a machine learning algorithm to choose the best allocation policy among Round-robin (RR), WaterFilling (WF), and Proportionally Fair (PF). However, using these policies alternately does not guarantee the best choice for the available set of users of a particular service requesting allocation; it only reduces the deficiencies that each one possesses by alternating between the three possible algorithms. For example, while RR is used, the network reproduces the failures of this method, negatively affecting the quality of service. The RR policy is a simple resource allocation approach that distributes the available resources equally among users in a cyclical sequence. It ensures that all users receive an equal share of the available resources, without favoring any particular one. However, as it distributes resources in a fixed and predictable manner, it may not be efficient in situations where some users require more resources than others, or when network conditions change rapidly. This can lead to inefficient resource utilization and degradation of overall network performance.

Faced with the challenge of efficiently managing radio resource allocation and aiming to minimize low-level human orchestration failures, this work proposes a solution based on reinforcement learning methods. The problem is formulated as a Markov Decision Process (MDP), in which a Radio Unit (RU), a network structure responsible for signal transmission/reception, needs to allocate Physical Resource Blocks (PRBs) among different users of network slices to perform the process in the network. Each slice corresponds to a specific set of users and services, such as eMBB, URLLC, and mMTC. The proposed algorithm uses Q-Learning to estimate an optimal policy that maximizes the network objectives of each service. The reinforcement learning agent makes dynamic decisions, choosing available users for allocation in the network considering the specific characteristics of each

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, FAPERJ grant SEI-260003/004771/2021, FAPESP grants 23/00673-7 and 23/00811-0, and CNPq grant 408255/2023-4. The authors also thank the OpenRAN@Brasil Program (MCTI Process No. A01245.014203/2021-14) and the Research, Development, and Innovation Scholarship Program from RNP.

slice. The resulting policy seeks to optimize the utilization of available resources, adapting to various scenarios of radio resource availability. By utilizing reinforcement learning techniques, this work addresses the limitations of conventional methods, providing an adaptive and efficient solution for radio resource management in Open RAN environments. Unlike complex algorithms such as Deep Q-Networks (DQN), which complicate resource analysis and orchestration, this approach emphasizes simplicity and adaptability. It enables the assimilation of strategies from both sophisticated and simpler solutions, enhancing overall performance and responsiveness for radio resource management in Open RAN environments.

The remainder of the paper is divided as follows: Section II explains the basic concepts of the work. Section III discusses related work. Section IV describes the considered scenario. Section V details the proposed reinforcement learning algorithm, presenting the environment modeling in Gym, a library for developing training environments for AI projects. The proposal evaluation occurs in Section VI. The conclusion and future research directions are discussed in Section VII.

II. BASIC CONCEPTS

Modern communication networks are under increasing pressure as the demand for connectivity and high-quality services continues to rise. Addressing these challenges demands techniques like network slicing and Artificial Intelligence (AI), which are developed and implemented to enhance network performance and meet the diverse needs of users and services. Network slicing allows for creating multiple virtual networks on a single physical infrastructure. Each slice is customized to meet the specific requirements of different services, such as bandwidth, latency, and security. For example, Enhanced Mobile Broadband (eMBB) provides enhanced bandwidth and higher data speeds, ideal for applications like high-definition video streaming, virtual/augmented reality, and online gaming. Ultra-Reliable Low-Latency Communications (URLLC) targets applications requiring highly reliable and low-latency communications, critical for real-time scenarios like vehicle-to-vehicle communication, remote machine control, and remote surgery. Massive Machine Type Communications (mMTC) facilitates efficient communication among many low-power, low-data-consumption devices, such as the Internet of Things (IoT) and remote monitoring.

Given the limitations of radio resources and bandwidth, intelligent allocation is essential to ensure efficient communication. Allocation policies, *i.e.* sets of rules or algorithms, determine how available resources are distributed among devices in the network, considering factors like bandwidth, transmission power, and other critical elements. These policies aim to achieve objectives such as maximizing network capacity, minimizing latency, or balancing load among network nodes. AI, particularly Machine Learning (ML), is essential in implementing these allocation policies. ML develops algorithms and models that learn from data. There are three main types of ML:

- 1) **Supervised Learning:** The algorithm is trained on a dataset that includes inputs and corresponding outputs, learning to map inputs to correct outputs and predict correct outputs for new, unseen inputs;
- 2) **Unsupervised Learning:** The algorithm is trained on a dataset without correct outputs, aiming to discover structures or patterns in the data, such as clusters or distributions;
- 3) **Reinforcement Learning:** The algorithm learns to take actions in an environment to maximize cumulative reward, receiving rewards or penalties based on the actions taken and their consequences.

The proposed work leverages these concepts to build an algorithm that understands the network environment and proposes a resource scheduling policy for users of different slices based on Reinforcement Learning.

III. RELATED WORK

The literature on Open RAN prominently features the application of ML techniques for resource allocation, network slicing, and policy scheduling. Arnaz *et al.* [1] provide a comprehensive analysis of the integration of intelligence and programmability in Open RAN, delving into the potential of disaggregation, virtualization, and the standardization of interfaces [1]. Their work summarizes the evolution of RAN technology and presents state-of-the-art technologies applied to Open RAN. Similarly, Azariah *et al.* discuss Open RAN-related projects, activities, and standardization efforts, further contributing to the field [4]. Couto *et al.* [5] enhance the research landscape by exploring public datasets that train deep learning models specifically for Open RAN [6]. Using existing datasets in mobile networks offers a robust foundation for training models tailored to Open RAN [6].

In resource allocation, Sharara *et al.* introduce a reinforcement learning-based approach to optimize the distribution of computational resources in Open RAN, focusing on fairness. Their method addresses the dynamic complexity of the environment by optimizing resource allocation between two services, URLLC and eMBB, improving the latency for URLLC without significantly compromising the quality of service for eMBB [7]. Bonati *et al.* [3] contribute further by developing an algorithm that selects the optimal policy for the network's current state and available radio resources. Their approach uses neural networks to determine when one of the three traditional scheduling policies, Round-robin (RR), Waterfilling (WF), or Proportionally Fair (PF), is most beneficial for network users [3]. Additionally, Thantharate *et al.* [8] emphasize the application of deep learning for efficient and reliable network slicing within 5G networks, introducing DeepSlice as a solution in this area [8].

The diversity of these approaches highlights the critical role and adaptability of ML techniques in contemporary Open RAN research. As these solutions are adapted to the Open RAN architecture, specific interfaces, and components can be explored further [4]. Table I summarizes the related work, comparing them according to the simplicity of the used

policy, the evaluated environment, the compliance to services requirements and the use of advanced ML techniques.

This paper complements previous investigations by focusing on a specific Open RAN dataset and developing a dynamic and straightforward network simulation environment and data analysis framework. It also proposes an efficient and versatile policy to mitigate the static limitations of simple policies for various service types. The proposed approach scales users optimally according to the network’s service objectives, prioritizing users based on the scenario rather than adhering to a rigid equality policy, thereby reducing failures associated with the deficiencies of simpler policies.

IV. ANALYZED SCENARIO

This paper proposes an algorithm for distributing radio resources among users for allocating network slices in a scenario composed of devices from different services statically connected to a Radio Unit (RU). In this network, the RU shares PRBs in a predetermined random proportion. PRBs are abstractions of network divisions, which in this context can be understood as frequency channel divisions used for signal reception and transmission. PRBs are allocated among the eMBB, URLLC, and mMTC services. The services have different requirements: eMBB aims to offer users high data rates to handle multimedia applications, enhanced mobility when users are moving, and high capacity to support numerous users; URLLC provides reliable communication with extremely low latency; mMTC seeks to facilitate efficient communication between devices (machine-to-machine), optimizing power consumption for moderate operation latency.

The OpenAI Gym simulator is employed to construct the scenario. Gym provides a consistent and easy-to-use interface for interacting with these simulation environments, allowing developers to experiment and evaluate machine learning algorithms. It supports the definition of observation and action spaces, rewards, episodes, and other essential functionalities for development and testing. In the simulated environment, a Radio Unit R is considered as the central object for training the orchestration method, i.e., for updating the Q-Table. The Q-table is a structure used in reinforcement learning methods to record and improve the effectiveness of actions taken in different network states. This table stores information about the quality or usefulness of performing specific actions in particular contexts. For each Open RAN Radio Unit (O-RU), a predefined and static set of users is defined for each provided service, symbolized by the service names, i.e., eMBB, URLLC, and mMTC. The total number of users is represented by n_users . Each O-RU is subdivided into three slices, one for each service, and each slice has its characteristics, including the number of available PRBs, $slice_PRBs$, the list of users currently requesting allocation, $user_request$, and the list of users that have already been allocated, $slice_users$. Users in the simulation are defined by a set of metrics, such as $packrate$, which is a basic statistic inspired by the dataset used for training and represents the packet transmission rate. Varying for services based on the work of Bonati *et al.* [3] for

eMBB users, the users produce a traffic rate of 100 packets per second (pkt/s). For URLLC, the users generate 125-byte packets with a rate of 10 pkt/s. Finally, in the case of mMTC, the users produce traffic consisting of 125-byte packets at a rate of 30 pkt/s. Additionally, a parameter called *window* is used, defining the time window that the allocated user will stay in the network, measured in seconds, being randomly defined in the interval [10-15]. It is during this period of time that the user generates the traffic. Furthermore, the variable $requested_prbs$ represents the amount of resources requested by users and is defined in the interval [3-5] at the time of user instantiation. This range of requested resource values is defined to not deviate significantly from the guaranteed amount of PRBs, which is used in the algorithm that distributes resources equally among users. The idea is that the user requests slightly more than what is needed, to accelerate the data transmission process in the network if allocated with the requested amount of resources.

At the time of user instantiation, the random variables *window* and $requested_prbs$ are generated by the network, but not always used directly, as the amount of PRBs allocated to each user may vary according to the policy in force in the network. Thus, it is necessary to proportionally recalculate the number of time windows in the network for users. For example, from the moment a user is allocated, the user generates packets at a rate of 30 pkt/s with 5 PRBs for a period of time of 2 *windows* (considering a window of 1 second) and, therefore, in total the user will send 300 packets, each one with 125 bytes. The algorithm is trained in window divisions, representing moments dedicated to user allocation. Hence, each new iteration of the algorithm is a new window with a period of 1 second. In each window, the network is updated, adding users who have requested allocation and deallocating those who have already completed the process in the network. User allocation is based on their metrics, and each user remains allocated for a calculated amount of windows in the network, allowing them to complete the entire data-sending process. The simulation environment in OpenAI Gym replicates the dynamics of an O-RU, considering different services and network slices, user metrics, and allocation windows. This environment is designed to be used in reinforcement learning algorithm training, providing a simplified yet realistic representation of the challenge of resource allocation in mobile communication networks.

In the context of this work, the Episodic Reinforcement Learning Model framework is employed for dealing with the inherent complexity of dynamic network environments. The algorithm is based on fundamental principles of the MDP, whose states, actions, and rewards are essential for decision-making. However, its implementation requires a detailed understanding of the interaction between the learning agent and the environment, as well as the development of suitable structures to guide the agent’s choices. Figure 1 shows, in a simplified way, class parameters representing the structure of the network environment. The Gym environment is composed of default parameters from the library. These

TABLE I
COMPARISON WITH RELATED WORK

References	Utilizes simple policies	Models a Network Environment	Meets Service Requirements	Uses advanced ML techniques
Arnaz <i>et al.</i> [1] Azariah <i>et al.</i> [4]	Yes	No	Yes	No
Sharara <i>et al.</i> [7]	No	Yes	Yes	Yes
Couto <i>et al.</i> [6]	No	Yes	Yes	Yes
Thantharate <i>et al.</i> [8]	No	Yes	Yes	Yes
Bonati <i>et al.</i> [3]	No	Yes	Yes	Yes
Popovski <i>et al.</i> [2]	No	Yes	Yes	Yes
This Work	Yes	Yes	Yes	Yes

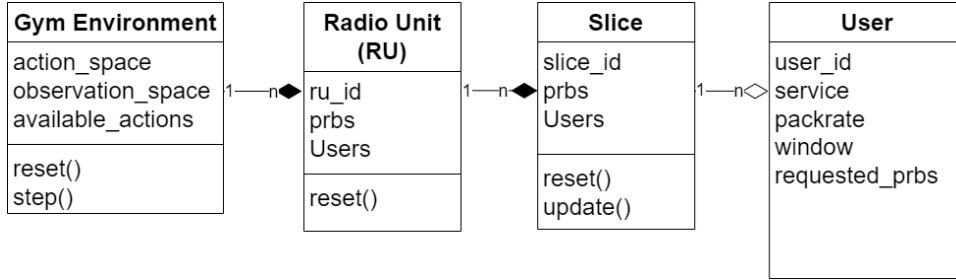


Fig. 1. Class Diagram of the Environment: Parameters and Methods

parameters are the `action_space`, representing the environment's action space initially representing all users of the service; the `observation_space`, being the set of training observations, the accumulated reward, and a list of currently allocated users; and the `available_actions`, being the available actions representing users who have requested allocation in the network. In the radio unit RU, the parameters are the `ru_id`, radio unit identification; `prbs`, the number of local PRBs of this network structure; `Users`, represents the users performing the process in the network. This list is dynamically updated during algorithm execution. Finally, the `User` class includes `user_id` and `service` identifiers for control purposes, along with key parameters that quantify the amount of traffic generated by each type of user. Regarding methods, `reset()` and `step()` are standard methods of any Gym environment, which are necessary to reset the training environment to an initial state for another execution. The `reset()` recursively calls the method of the same name of related classes, while `step()` is the main function of the algorithm, being called for each step of the environment simulation.

V. REINFORCEMENT LEARNING MODELING

In the model proposed in this work, the size of the initial state space is defined by the product of the factorials of the number of users in each category: $|users_{eMBB}|! \times |users_{URLLC}|! \times |users_{mMTC}|!$. Here, $|users_{eMBB}|$, $|users_{URLLC}|$, and $|users_{mMTC}|$ denote the cardinalities of the sets of users in the respective cate-

gories. This expression denotes all sequences of users allocated in the network, where $|user_{service}|!$, and $service = eMBB, URLLC, mMTC$, represents all possible combinations of user allocation for the respective service. The space decreases with each iteration since a user cannot be allocated twice in the same episode. A list of allocated users will be the environment's state, representing the current network organization. Thus, a state is defined as

$$\text{State} = (users_{eMBB}, users_{URLLC}, users_{mMTC})$$

During a window w at step i , one user will be selected for each slice. These users will remain in the network for several n windows that vary according to the amount of PRBs allocated to each user. The variable `slice_prbs` represents the number of PRBs available to users in their respective services. The action `action`, at each step i , is to select a user who requested allocation in the slice. Initially, the action space is the same as the state space and thus does not consider users who have already been allocated or are currently allocated in the network. An allocated user remains in the network until the process finishes and, at the end, returns the used PRBs to the resource count. The action space is mathematically defined as

$$\text{Action Space} = (users_{eMBB}, users_{URLLC}, users_{mMTC})$$

The reward for executing an action is intrinsically related to the type of service and its objectives. In this work, the reward function for each type of service is defined as:

VI. SIMULATION AND RESULTS

- eMBB: Maximize Throughput
- URLLC: Maximize transmitted packets
- mMTC: Minimize buffer queues

The eMBB service aims to maximize users’ transmission rate (*bitrate*). URLLC is a service type that prioritizes low latency; therefore, its reward includes maximizing transmitted packets, which will allow the users to experience a minimum delay. Given a traffic rate (*packrate*), mMTC seeks to minimize buffer queues. This is because mMTC is a service that values energy consumption, and repeating the same action wastes energy. The proposed algorithm adopts Q-Learning as a reinforcement learning method to guide the agent’s decisions in optimizing network services. Q-Learning utilizes the Bellman equation, which describes how the Q-values are updated over training time based on the state, reward formulation, and performed action. The Bellman equation used for updating Q-values takes the following expression

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot \left(r + \epsilon \cdot \max_{a'} Q(s', a') \right) \tag{1}$$

In this equation, $Q(s, a)$ represents the Q-value for state s (state) and action a (action); α is the learning rate that controls how quickly new knowledge is incorporated into the Q-table, indicating how much the knowledge gained from performing an action affects the accumulated knowledge from previous actions; r is the reward received by the agent for taking action a in state s ; ϵ is the discount factor that determines the importance of future rewards; s' is the next state resulting from action a ; and a' is the next action chosen by the agent in state s' . Individual Q-Tables (Q_table) are created for each service. The table updates reward values as the environment is trained according to the Bellman equation. The Q_table has dimensions of $n_{users} \times n_{users}$ for each service, where each row represents the last chosen user, and each column represents the next user resulting in the maximum reward for the objective at a particular step of the algorithm, for example, the last user chosen will index the row, so considering that the past action was the best possible (user index = 2), according to the algorithm, the next user who will maximize the reward will be represented by the column (another user, with index 3 for example), forming the ordered pair that best benefits the network in this situation. During each training episode, the agent bases its actions on these tables, following the principle of exploration through the current values of the table’s users. In other words, at each `step` of the environment, the algorithm searches the table for the user that maximizes the reward based on the current state of the network. Then, the Q-Tables are dynamically updated based on the rewards obtained by the agent and estimates of future rewards. These updates allow the agent to learn to make more efficient decisions in similar future states.

The performance of the Reinforcement Learning (RL) algorithm, called DQL (Deep Q-Learning) in the results, is evaluated through simulations using OpenAI Gym and the Python environment, utilizing libraries such as NumPy. After the training phase and the values in the Q_Table converge, the network changes the users, with attributes similar to those of the training, but with a certain variety for the ones that were used on the tests, 4096 epochs are used, both for training and testing. The proposal’s performance is compared to that of a Round-robin (RR) resource allocation method. The RR algorithm is a simple resource allocation method that evenly distributes resources sequentially among users in a repetitive cycle. It is assumed that the O-RU has a total of 35 PRBs, which are distributed to the slices of users for the eMBB, URLLC, and mMTC services. The distribution of these resources follows Table II, which describes different situations of available PRBs for each service, using a random distribution with a minimum value set as 7 PRBs to ensure that the sum of all PRBs is 35. Each service has five possible users for the training process, with different users used for training and testing the algorithm. With each new episode, a new organization of PRBs is made to promote dynamism in training.

TABLE II
TABLE OF DISTRIBUTIONS OF PRBS USED

Modes	eMBB	URLLC	mMTC
1	16	10	9
2	10	11	14
3	14	11	10
4	11	16	9
5	18	7	10
6	13	14	8
7	15	10	10
8	16	8	11

The simulation dynamics are conducted by a custom RL environment formulated using OpenAI Gym, reflecting the problem conditions in a simplified manner. During each episode, a distribution of PRBs among the services is used according to Table II. The RL agent makes decisions based on observations, which contain a list of users allocated at the moment at the network, the value of idle resources, and a user requests list of the environment. After the training phase and the values in the Q_Table converge, the tests are conducted by modifying the users of the environment. Thus, it is possible to validate the efficiency of the algorithm without biasing the training data. The performance metric used is the reward of the allocation algorithms, as the training metrics. Furthermore, we also measure latency, queue latency and throughput of all users. The exploration parameter ϵ is adapted through the episodes to balance the exploration of the action space, decaying at a rate of $\frac{\epsilon}{episodes}$. The learning rate is adjusted to 95% to optimize convergence and the RL agent’s performance. For visualization proposes, Figures 2(a), 2(b), and 2(c) show the average reward during tests, for eMBB, URLLC, and mMTC, respectively. The figures show that for rewards that

maximize metrics (eMBB and URLLC), for many iterations, when users stay longer on the network, the proposed algorithm is superior to RR, stabilizing at higher rewards. In the case of mMTC, which aims to minimize buffer queues, the discounts are smaller for the proposed algorithm when the time required for users to transmit their data over the network increases.

Figure 3 aims to evaluate the overall network performance based on the iterations required for the algorithm to allocate the entire user set. Each episode has the same network configuration for both policies used. From these results, it can be inferred that the proposed method using DQL takes about 20% fewer iterations (each iteration has 1 second) to complete the same episode compared to the RR policy.

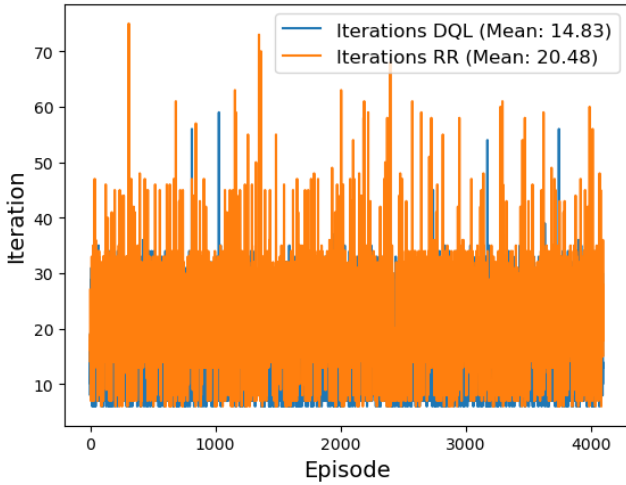
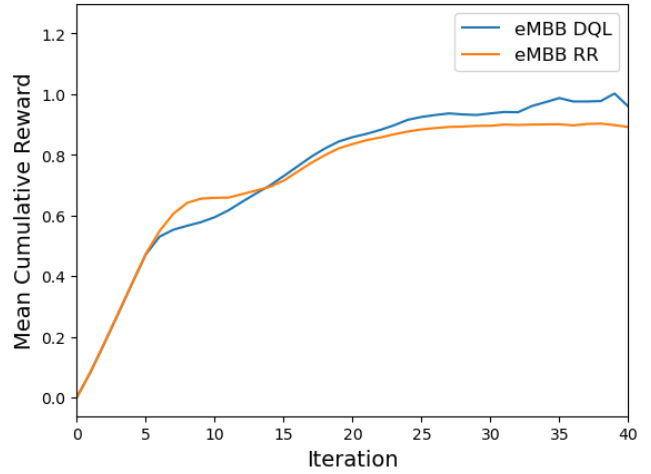
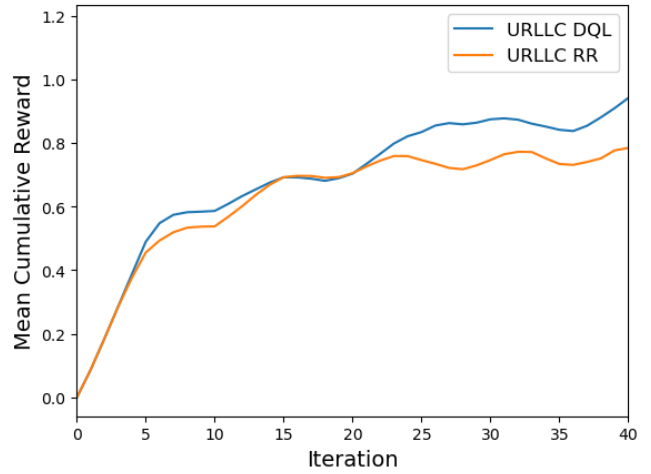


Fig. 3. Number of iterations per test epoch.

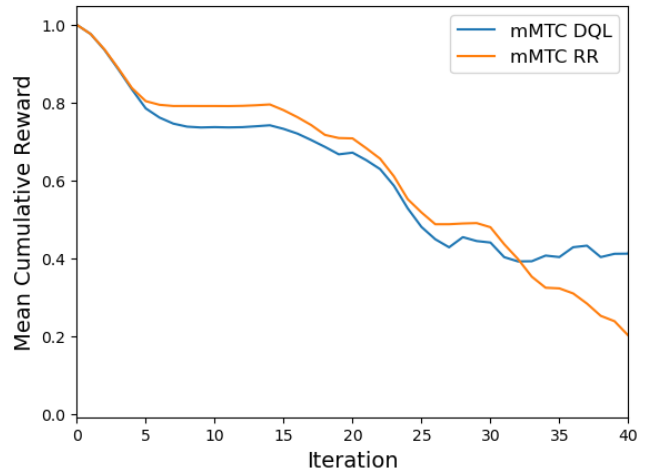
Latency is also evaluated, which is the total time the user remains allocated in the network, and the algorithm's queue time, which is the waiting time for the user to be allocated from the moment it requests allocation. Figures 4(a), 4(b), 4(c) represents the latency, and Figures 5(a), 5(b), 5(c) shows the average queue waiting time for each user in the slice over 4096 epochs run during the simulation. For this total set of simulations conducted, users allocated with the RR policy and the proposed DQL reinforcement learning algorithm achieve, respectively, 14.83 and 20.48 iterations for all users in the test set that is shown in Figure 3. In the overall network latency, the DQL algorithm keeps users allocated 35% fewer iterations than RR, as the proposed method manages resources more efficiently, allowing users to transmit or receive data through the network more quickly. In the queue latency, there is a noticeable improvement of 37% in iterations, as resources are being better distributed, increasing the chances of a user requesting allocation when the necessary PRBs are available in DQL, this metric is especially important for mMTC users to achieve their service requirements. Analyzing the eMBB throughput in general, Figure 6(a), DQL improves the transfer rate of users by 50%, except for user 4. This algorithm behavior is due to the equal distribution of resources to users. The proposed algorithm aims to maximize the



(a) eMBB.



(b) URLLC.



(c) mMTC.

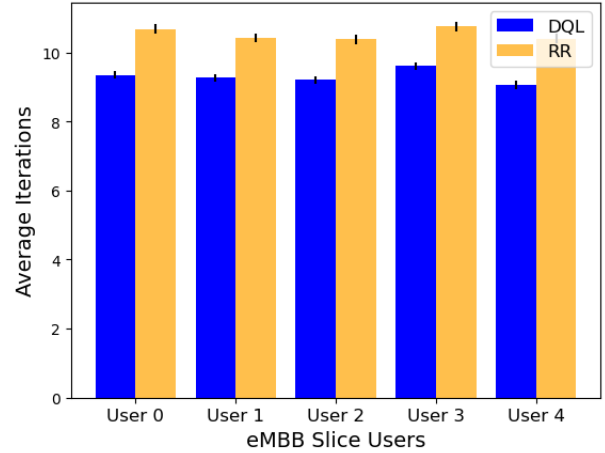
Fig. 2. Cumulative Reward during the tests for each service.

throughput, and in the simulation, the algorithm's choices benefit other users more at the expense of reducing the rate

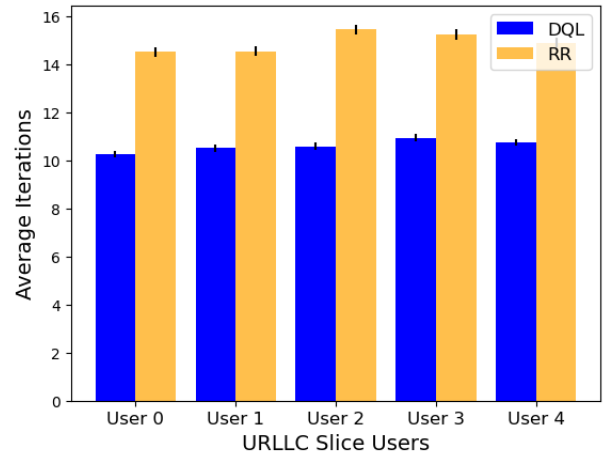
of one of them. In this case, User 4 had his transfer rate reduced by 50% on average. It is worth noting that the improvement is significant for all service types. Although the algorithm primarily focuses on network objectives, it does not completely abandon equality among users. In the case of URLLC, Figure 6(b), RR is better than the solution proposed in Reinforcement Learning since the algorithm focuses on maximizing the packets transmitted at the cost of reducing the throughput. For mMTC, Figure 6(c), users are allocated in a similar way, which indicates that for this type of service, throughput is not a factor that effectively improves service requirements, as the likely decrease in resources given to users and optimized choices will allow the number of PRBs given to each user, so it maintains the same values.

VII. CONCLUSION

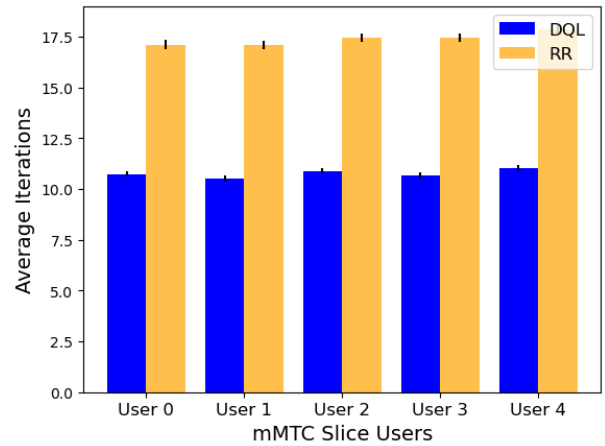
The challenge of efficiently allocating radio resources in Radio Access Networks (RANs) can be addressed by new solutions due to the innovations brought by Open RAN architecture. Faced with the increasing demand for various services such as eMBB, URLLC, and mMTC, efficient resource management is essential to optimize network performance. The solution proposed in this article focused on the development of a mechanism based on reinforcement learning techniques, specifically the DQL algorithm, for efficient orchestration of network slices for various services. By considering the complexity of the requirements of each service type, the DQL model demonstrated adaptability and effectiveness in dynamically allocating resources. The development of the reinforcement learning algorithm took into account the specific characteristics of each network slice, reflecting the needs of each service. Comparative analysis with the traditional round-robin (RR) allocation method highlighted the superiority of the reinforcement learning model in terms of resource utilization efficiency and adaptability to changes in network conditions, especially for eMBB service. Simulation in the Python environment, using the OpenAI Gym library, allowed for evaluating the performance of the proposed mechanism under controlled and realistic conditions based on RAN scenarios. Latency evaluation showed how each user's latency is affected by the policies used, and through an average of iterations, there was a reduction of over 20% for the proposed algorithm (DQL) compared to the simple policy (RR). Similarly, there was a 35% reduction in waiting time (queue latency) for allocation and a 20% increase in user throughput. As suggestions for future work, expanding the study to more complex scenarios, such as user mobility environments and variations in traffic load, could be considered. Additionally, exploring advanced reinforcement learning techniques and optimizing hyper-parameters could further enhance the algorithm's performance in dynamic environments. Furthermore, federated learning methods could be used to share experiences acquired by each radio unit to distribute the training cost and absorb data from different regions with different characteristics and users.



(a) eMBB.

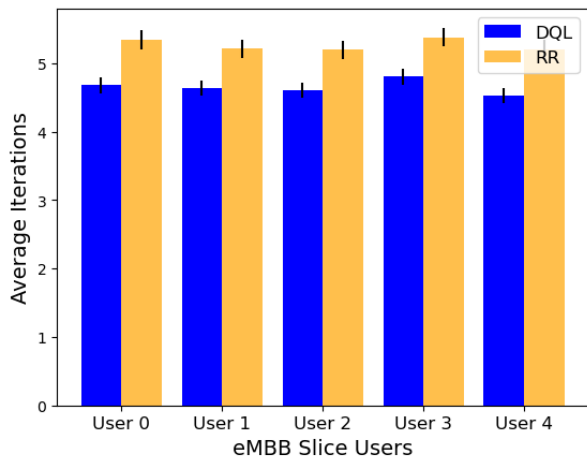


(b) URLLC.

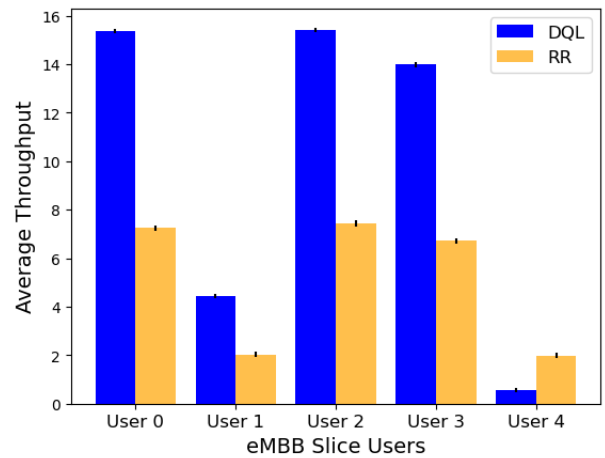


(c) mMTC

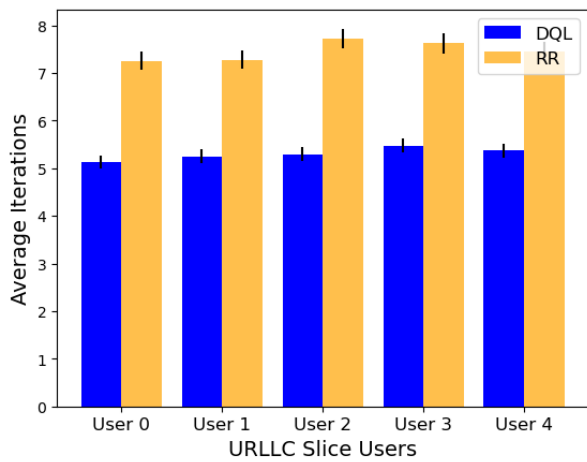
Fig. 4. Latency of services' users during tests.



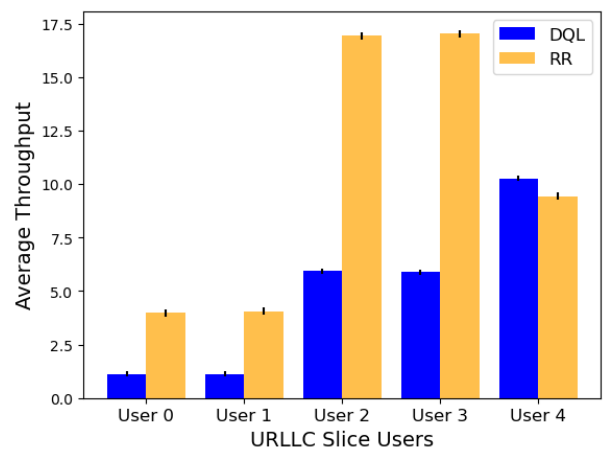
(a) eMBB.



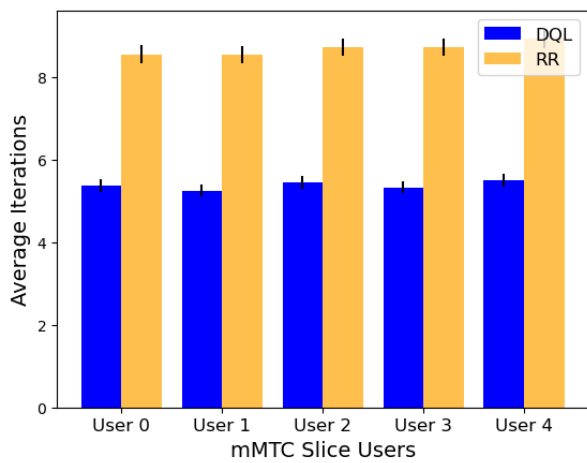
(a) eMBB.



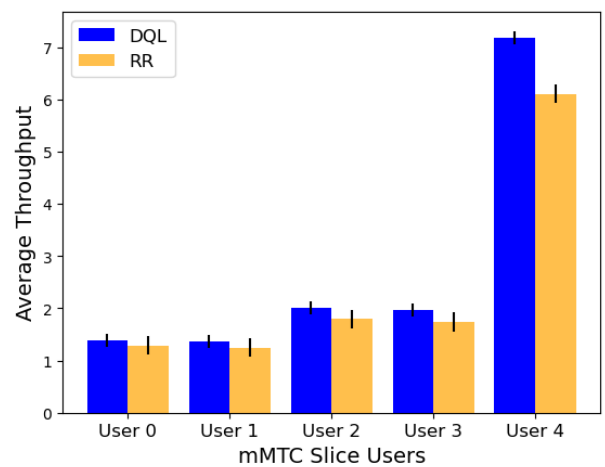
(b) URLLC.



(b) URLLC.



(c) mMTC.



(c) mMTC.

Fig. 5. Queue Latency of services' users during tests.

Fig. 6. Average throughput of services' users in the network.

REFERENCES

- [1] A. Arnaz, J. Lipman, M. Abolhasan, and M. Hiltunen, "Toward integrating intelligence and programmability in open radio access networks: A comprehensive survey," *IEEE Access*, vol. 10, pp. 67 747–67 770, 2022.
- [2] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View," *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.
- [3] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in o-ran for data-driven nextg cellular networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.
- [4] W. Azariah, F. Bimo, C.-W. Lin, R. Cheng, N. Nikaein, and R. Jana, "A survey on open radio access networks: Challenges, research directions, and open source approaches," *Sensors*, vol. 24, p. 1038, 02 2024.
- [5] R. S. Couto, P. Cruz, M. E. M. Campista, and L. H. M. K. Costa, "Using public datasets to train o-ran deep learning models," in *2st International Conference on 6G Networking (6GNet)*, Paris, France, Oct 2023.
- [6] R. S. Couto, P. Cruz, R. G. Pacheco, V. M. S. Souza, M. E. M. Campista, and L. H. M. Costa, "A survey of public datasets for o-ran: fostering the development of machine learning models," *Annals of Telecommunications*, pp. 1–14, 2024.
- [7] M. Sharara, T. Pamuklu, S. Hoteit, V. Vèque, and M. Erol-Kantarci, "Policy-gradient-based reinforcement learning for computing resources allocation in o-ran," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, Paris, France, Nov 2022.
- [8] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5g networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2019, pp. 0762–0767.