This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

Design of Scalable Population of Reinforcement Learning Agents for Autonomous 5G Radio Link Control

John Cosmas, Kareem Ali, Ali Mahbas
Prince Kwaku Boakye, John Miguel
Brunel University London
Uxbridge, Middlesex
john.cosmas@brunel.ac.uk

Victor Gabillon, Alexandre
Kazmierowski, Lewis Sear,
Thales SIX GTS,
Gennevilliers (Hauts-de-Seine), France

***Abstract*— This research demonstrates how MATLAB's Reinforcement Learning Markov Decision Process (MDP) Example Model can be used to design Radio Link Control MDP Reinforcement Learning (RL) agent. Since the number of agents in MATLAB's RL toolbox is not scalable beyond one agent, then an agent scalability scheme is required to design RL agents in MATLAB's RL toolbox and then realize multiple lightweight simultaneously operable Python instances of it for each of the multiple user equipment UE in a network.**

Keywords—MATLAB Reinforcement Learning Tool Box, 5G Autonomous Radio Link Control Scalable population of Reinforcement Learning Agents

I. INTRODUCTION

The goal of this research is to first train a Markov Decision Process (MDP) radio link control model Reinforcement Learning (RL) agent for 5G Radio Link Control using MATLAB's Reinforcement toolbox by building on its RL MDP Example Model [9] and then secondly use the Thales TheRLib MA-DRL Tool [1] to produce many light weight Python instances of these agents so that they can be simultaneously applied to multiple user equipment instances (UE) in a 5G Cell-Free Network (CFN) model.

MATLAB's Network Topology Visualisation of Intercell Interference Model as shown in Figure 1 was used as a suitable CFN model.

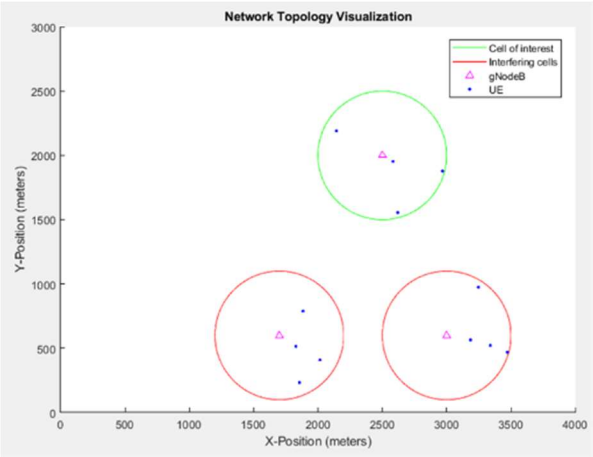

Fig. 1. MATLAB's Network Topology Visualisation of Intercell Interference

This model analyses the interference experienced by UE in the target green cell due to the effect of simultaneous transmissions on the same carrier frequencies of the adjacent red Cells, as shown in Figure 1. By assuming that all the UEs have been clustered to their nearest gNB this can be considered as a suitable CFN model [12].

The objective of this research is to train the RL agents for each of the twelve User Equipment (UE) in the CFN to select Modulation and Coding Scheme (MCS)/ Channel Quality Indicator (CQI) to maximise their throughputs. The problem is that the number of agents in MATLAB's RL toolbox is not scalable beyond one agent. Therefore, an agent scalability scheme is required to design RL agent in MATLAB's RL toolbox and then realise multiple lightweight simultaneously operable Python instances of it for each of the 12 UEs.

This is an example of independent learning category of Multi Agent Reinforcement Learning (MARL) [17], which if combined with slicing resource allocation scheduler may require to be enhanced by applying cooperative or competitive agent training and execution approaches.
MATLAB's examples of RL applications are not so easy to be applied to any other application other than the one it has been designed for because, while the main program GenericMDPExample.m is easy to edit, there is no easy way of overriding the important move_() function in the AbstractMDP.m file, adding more simulation parameters in the GenericMDP.m file and modifying SeriesTrainer.m for providing a visual feedback of the episode number and for starting at the best state at start of each episode, since these files are all part of the MATLAB system files, which under normal circumstances do not have write permission to modify. We overcame this constraint by installing MATLAB on our own Linux workstation, where we could modify write permissions.
The architectural impact of a scalable population of RL Agents is they can be trained from dataset on MATLAB RL toolbox and deployed to Open Radio Access Network (ORAN) and multiple UEs thereby producing a workflow for training autonomous networks.

II. MODULATION & CODING SCHEME & CHANNEL QUALITY INDICATOR

A. Existing RLC Scheme

In the 5G Radio Access Network (RAN) design, the MCS is a key mechanism that controls how many meaningful bits may be conveyed by a single Resource Block (RB) in the wireless communication channel. The modulation scheme and the coding rate are two essential components that the MCS establishes. The coding rate affects the ability of the transmitted data to adjust for errors, whereas the modulation technique determines how information is encoded into the carrier signal. The right MCS must be chosen in order to maximise data flow and achieve dependable communication [11] of which table 5.1.3.1-1 in this reference shows MCS index and their corresponding spectral efficiency. Higher MCS levels provide higher data rates, but they also call for more favourable channel conditions [2]. The higher the MCS the greater the throughput but also the greater the probability

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

of a Transport Block Error (BLER). If a BLER occurs then the whole Transport Block is discarded and a throughput is zero with request for retransmission made using the Hybrid Automatic Repeat Request (HARQ).

In conventional operation, 5G RAN uses the CQI to dynamically modify the MCS to suit the various channel circumstances that various UEs may encounter based on heuristic equations [14] and look-up tables [15] that have been characterised and pretrained from network simulations [15]. This does not always select the most optimum MCS. Adopting a RL RLC solution, which continuously trains itself, could potentially select the most optimum MCS.

B. *Proposed New RL RLC Scheme*

The principle of operation for RL downlink transmission is that each UE records its assigned MCS at the MAC layer and evaluates the successful outcome of its assigned Transmission Block as a Packet Data Unit (PDU) using Cyclic Redundancy Check (CRC) from the HARQ ACK/NACK in the PHY layer and then based on this outcome the UE decides to change the MCS by increasing/decreasing/remain-same in order to maximise Throughput/Efficiency and notifies its gNB, as shown in Figure 2. The merit of this approach is that the UE decides for itself the optimum MCS to maximise throughput.

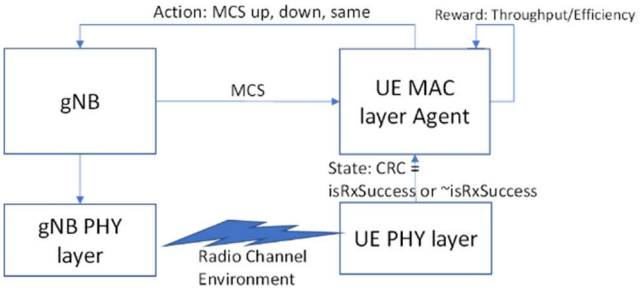

Fig. 2. Reinforcement Learning control RAN Link

III. THERLIB MULTI-AGENT DEEP REINFORCEMENT LEARNING (MA-DRL) TOOL

In the DRL control of the RAN links, the UE decides on its action to change the MCS to support its needs based on its sensing of the state of the environment based on MATLAB's Intercell Interference Model.

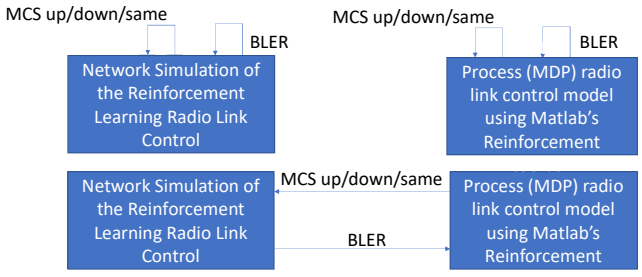

Fig. 3. Relationship between Network Simulation, MATLAB's MDP Model

How MATLAB Network Simulation of the RL Radio Link Control relate to the MDP Radio Link Control model using MATLAB RL toolbox is illustrated is Figure 3. Each was modelled independently with internally generated MCS up/down/same actions and BLER result. These need to be integrated with each other with 12 instances of the MATLAB MDP for each UE. The top half of the figure shows the two software as separately developed simulations. The bottom half of the figure shows how these two softwares are integrated.

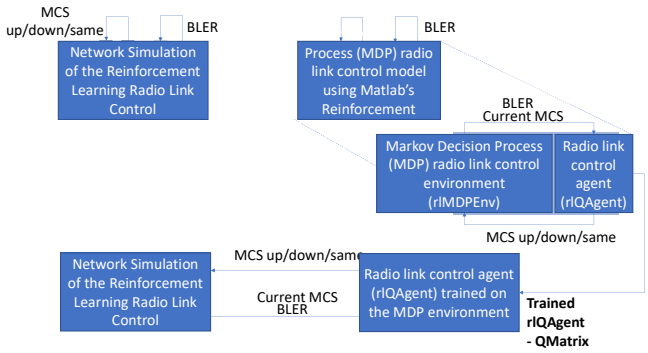

Fig. 4. Relationship between Network Simulation, MATLAB's MDP and TheRLib MA-DRL platform

The high-level architecture of a pure-MATLAB integration is as shown in Figure 4. The " MDP radio link control model using MATLAB's RL toolbox" is broken down into the "environment" and "agent" objects that are used by the MATLAB RL toolbox and the trained "agent" is used for with the Network Simulation. The small-dimensional agent trained on the MDP model, which is leveraged to cope with performance and scale because 12 instances of the agent, is required for each of the 12 UE in the Network Simulation.

TheRlib can also be used to train and deploy the agent using MDP as shown in Figure 6.

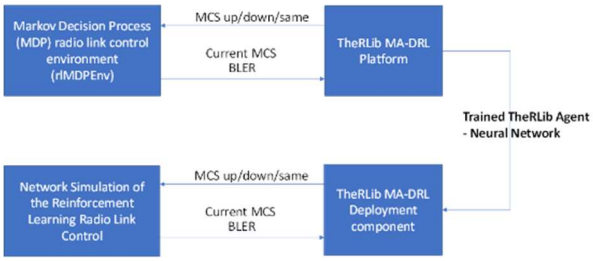

Fig. 5. Applying TheRLib MA-DRL platform for both training and deployment of MDP

Note that TheRLib interacts with the MDP "environment" and manages the training process instead of the MATLAB RL toolbox. When interacting with the Network Simulation, the "TheRLib MA-DRL Deployment component" code is in a simple script that loads a serialized TheRLib agent from a data file, instantiates the MATLAB RL environment and performs the agent-interaction environment. The merit of this approach is that an instance of TheRLib agent has a much smaller footprint and computes much more efficiently when using higher-dimensional states than the MATLAB RL toolbox version of it, so can be used to train and deploy many of them such as the 12 UEs required in the network simulation.

TheRLib can be used to train and deploy an agent to the MDP Environment shown in Figure 6. The agents can either be Actor-Critic Networks trained with algorithms such as soft actor-critic or Deep Q-Networks trained with Q-learning.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

Here the merit is that multiple states can be incorporated such as for example MCS/CQI states for Radio Link Control and RB Capacity states for a Slice Capacity Control system. Training can be performed with the MDP radio link control environment, or a refined version with more complex states, if it provides the RL environment interface, i.e., step and reset functions.

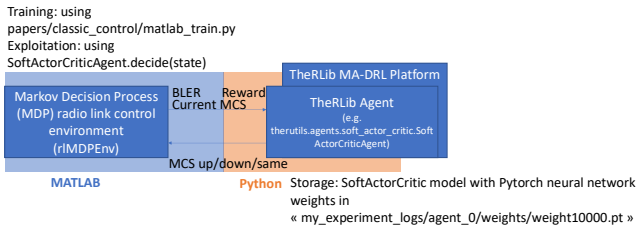

Fig. 6. Applying TheRLib MA-DRL platform for Neural Network training and deployment

The interface to The RLib tool is shown in Figure 7 where those agent successfully trained into Python code agents are shown in green whilst those unsuccessfully trained in dark blue.

Fig. 7. User Interface to The RLib tool

IV. RL MDP 5G Radio Link Control MATLAB Development Environment

A. Markov Decision Process (MDP) approach

The MDP approach in RL for the 5G RAN setup entails characterising a series of states, actions, transition probabilities, and rewards. The RL's chooses to increase RAN performance are included in the action space. The possibility of changing to a new state depending on activities is shown by transition probabilities, which reflect environmental uncertainty. State-action combinations are given numerical rewards via the reward function, which directs the agent towards more advantageous RAN configurations [10].

When applying RL to an MDP, the RL agent discovers the best course of action through interactions with the environment. For the purpose of making wise judgements, the agent experiments with various activities, monitors results (rewards), and updates its knowledge. The agent develops methods to lessen interference, efficiently manage resources, and improve overall RAN performance over time. By using an adaptive method, the agent may configure the 5G RAN in a way that is effective and intelligent and improves wireless communication capabilities while dynamically responding to changes in the wireless environment [3].

B. Adaptive MCS Selection using MDP with Block Error Rate

The reinforcement learning-based methodology was used to integrate MCS, CQI, and BLER. The MDP environment, which models the 5G RAN setup, was interacted with by the reinforcement learning agent. In order to choose the proper MCS levels for each UE in the system, the agent obtains knowledge from the CQI feedback and BLER data. The agent seeks to maximise cumulative rewards by optimising the MCS selection based on real-time channel circumstances, which equates to obtaining greater data rates while ensuring dependable and error-free communication [2].

The core of our approach lies in the move_function, which simulates state transitions in the MDP. The function accepts an action, representing the MCS to be used, and returns the next state, the received reward, and a flag indicating whether a terminal state has been reached. Here, the reward R is set to zero if a block error occurs.

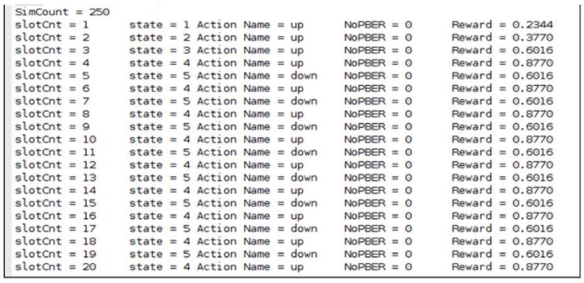

Fig. 8. MCS,CQI & BLER

If no block error has occurred then then the training episode is not terminated early and is allowed to continue for the whole frame of 20 slots when it is terminated, which is shown in the Figure 8. Whereas if a single block error has occurred then the training episode is terminated, since less than 1 in 20 block errors or < 5% is allowed. Less than 2 in 20 block errors or < 10% is allowed then 2 block errors must have occurred for the training episode to be terminated. The methodology for developing an agent in a MDP environment in MATLAB can be summarized into below four key steps and their sub-tasks.

1. Creating the MDP Environment
2. Creating the Q-Learning Agent
3. Integrate the MDP Environment with Q-Learning Agent
4. Training and Validating the Q-Learning Agent

1. CREATING MDP ENVIRONMENT

The first step in working with a MDP involves establishing the fundamental elements that define the environment in which the agent will operate. This foundational step has several sub-steps:

a) DEFINING THE STATES

The createMDP function in MATLAB is used to construct an MDP model in the initialisation phase. This function allows the user to specify the number of states and actions required for the MDP framework. For the purposes of this project, the MDP is designed with 15 states that correspond to the 15 CQI indexes and their corresponding MCS states and three possible actions: 'UP', 'DOWN'; and 'SAME'; These actions signify the choices available to the RL agent at each state. In this context, each state within the 5G RAN is representative of a specific configuration or situation [3].

b) DEFINING THE STATE TRANSITION MATRIX

The state-action transition matrix (MDP.T) is a three-dimensional matrix where each element MDP.T(i, j, k)

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

specifies the transition probability from state 'i' to state 'j' when taking action 'k' [3].

In our model, the transition probabilities were deterministic. For instance, a transition probability of MDP.T(1,2,1) = 1 indicates that if the system is in state 1 and the action 'UP' is taken (indexed as 1), the system will transition to state 2 with a probability of 1. Actions are indexed as 1 for "UP," 2 for "DOWN," and 3 for "SAME." States are indexed numerically starting from 1.

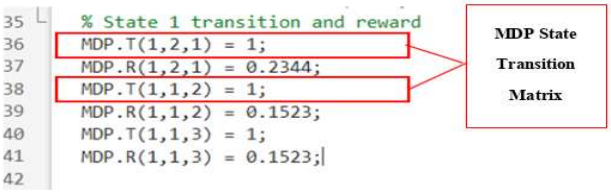

Fig. 9. Specifying the Transition Matrix

c) SPECIFYING THE REWARD MATRIX

The reward matrix (MDP.R) is another three-dimensional matrix where each element MDP.R(i, j, k) specifies the reward obtained when transitioning from state 'I' to state 'j' through action 'k' [3]. For example, MDP.R(1,2,1) = 0.2344 implies that a transition from state 1 to state 2 through action 'UP' would yield a reward of 0.2344. The reward is the spectral efficiency defined for a MCS as defined in table 5.1.3.1-1 in reference [11]. The reward matrix, which gives the RL agent rapid feedback for each state-action transition, is essential component of the MDP environment. It shows the benefits the agent gains from doing particular actions and changing states. The agent's behaviour may be influenced and made to make choices that result in greater rewards by altering the reward matrix.

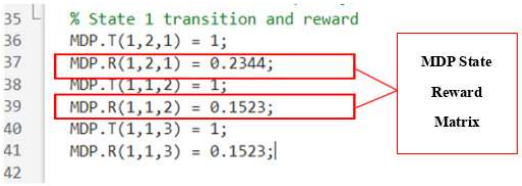

Fig. 10. Reference Simulation for Reward Matrix

d) INCORPORATING BLER IN THE MDP STATES

In our MDP model for optimizing 5G RAN configuration, we incorporate the BLER as an essential state-dependent parameter, that constitutes our dataset until such point the Network Simulation can be used for providing BLER for each UE. It displays the likelihood that a block of sent data (in a slot) will be incorrectly received. The MATLAB code assigns BLER values to each of the 15 states in our model, stored in the array MDP.E.

- Low BLER States: The states with MDP.E values of 0.01 represent configurations or conditions where the BLER is very low, indicating high reliability.
- Moderate BLER States: The states with MDP.E values around 0.5 to 0.9 signify moderate to high levels of block errors, possibly requiring corrective actions or reconfigurations.
- High BLER State: The state with an MDP.E value of 1.0 is an extreme case where every block is erroneous and likely signifies a severe issue requiring immediate attention.

These BLER values are used to guide the decision-making process of our reinforcement learning agent.

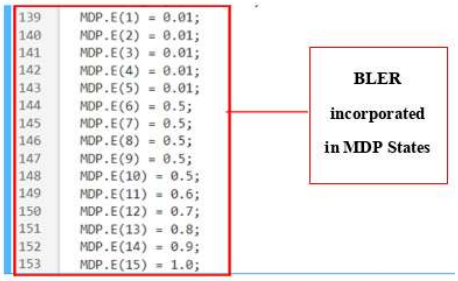

Fig. 11. Reference Simulation for Block Error Rate

e) DEFINING STATE SPACE

Initially the agent was configured to always start at state 1 but subsequently modified to start from the best state as identified by the largest reward recorded in the QMatrix to ascertain if performance in training the agent is improved with this modification. This was realised by modifying in top level RLCReinforcementLearning.m file function

```
env.ResetFcn = @() 1;  or  env.ResetFcn = @() MDP.StartState;
```

and inserting in SeriesTrainer.m choice of best state:

```
if any(QTableMatrix)  % if Qmatrix is not all zero
    hvalue = max(QTableMatrix(:,2))
     location = find(QTableMatrix(:,2) == hvalue)
    env.ResetFcn = location
end
```

One and two state transitioning, as illustrated in Fig 12 and Fig 13 respectively, was explored to determine the advantages/disadvantages of each approach.

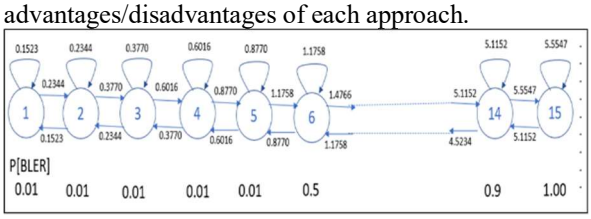

Fig. 12. MDP against BLER [4]

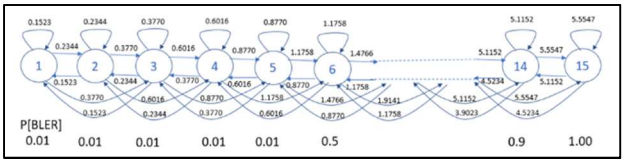

Fig. 13. MDP against BLER for up to two state transitioning[13]

f) CONTINOUS DECISION MAKING WITHOUT TERMINAL STATES

In our study on 5G RAN RLC configuration using reinforcement learning, we have designed MDP model that intentionally lacks specific terminal states. This design choice reflects the continuous operational nature of radio link control network management task in a 5G environment. A training episode duration of a Radio Frame or in our configuration for 20 slots.

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

2. CREATE Q-LEARNING AGENT

In the realm of reinforcement learning, a Q-Learning Agent aims to learn the optimal policy to navigate through a given MDP environment. By iteratively updating a Q-table based on rewards and state transitions, the agent learns to make decisions that maximize cumulative rewards over time [5].

3. INTEGRATION OF MDP ENVIRONMENT WITH Q-LEARNING AGENT

Finding the best policy in an MDP environment may be done according to the agent's experiences, the Q-learning algorithm repeatedly updates the Q-values, which are assessments of the quality of each action-state combination. As part of the integration, Q-value updates are made while learning is taking place utilising the state transition and reward matrices specified in the MDP environment [7][8].

The BLER values serve as the reward, the probability of BLER matrix serves as the environment, and the UE, operates as the agent in the RL cycle [16]. The CQI values reflect the state, while the BLER values represent the Reward. Choosing the appropriate MCS based on the relevant CQI is the agent's activity [6].

4. TRAINING AND VALIDATING THE Q-LEARNING AGENT

The success of a Q-Learning Agent hinges on effective training and validation. In the training phase, the agent interacts with the environment to update its Q-table, thereby learning the optimal policy to maximize cumulative rewards. Subsequently, the validation phase tests the agent's learned behaviour to ensure its efficacy and reliability [5].

V. IMPLEMENTATION OF STATE TRANSITION LOGIC ABSTRACT MDP

In our RL implementation, the move_ function plays a crucial role. Originating from an abstract class AbstractMDP, this function can be modified or overridden to implement the specific logic for MCS selection. The function takes in the current state and action as inputs and returns the new state, reward, and a flag indicating whether the simulation has reached a terminal state.

The three additional parameters required for this modified move_() function is

1. Transport BLER to signify if a Transport Block has experienced an error or not.
2. Probability of an error E occurring in any MCS state
3. Slot Count slotCnt that counts the 20 slots in a frame and terminates the frame once the 20th slot has been transmitted.

Initialization:

At the beginning of each call to the move_ function, certain initialization steps are undertaken. The time slot count (obj.slotCnt) is incremented by one to keep track of the number of steps taken in the environment. The initial reward R is set to zero, and the current state S0 is fetched from the object's property.

Terminal State Check:

The function first checks if the current state is terminal using the helper function isTerminalState(obj). In the context of a terminal state, the episode concludes, and the agent starts a new episode.

State Transition Logic:

The next state is calculated based on a pre-defined state transition matrix. A random number generator is used in conjunction with this matrix to determine the next state probabilistically.

Reward Calculation:

The reward for the transition is fetched using another helper function getRewardTransition_(obj, ActionName) which likely refers to a reward matrix that specifies the reward for each state-action pair.

Simulating Block Error and Its Effects:

The function simulates the occurrence of block errors. A random number is drawn and compared against a block error rate specific to the current state. If the random number is smaller, it simulates the occurrence of a block error, resetting the reward to zero and incrementing a counter for the number of block errors (obj.NoPBER).

Logging and Debugging:

For tracking and debugging purposes, crucial information is printed at each step. This includes the time slot, current state, action taken, total number of block errors, and the reward.

Updating the Environment:

The state of the environment and the termination flag are updated based on the logic and counters discussed above.

VI. RESULTS AND ANALYSIS

A. Result overview for 1 state transitioning

Our specific RL model, embodied by the Episode Manager, utilizes an RL agent named 'rlQagent' operating within a MDP Environment ('rlMDPEnv').

The agent's task is to dynamically configure the network's MCS, an essential aspect of 5G RAN that has far-reaching implications for network latency, throughput, and reliability.

The 'Episode Reward' on the y-axis serves as a quantitative measure of how well the rlQagent is performing its task of 5G RAN configuration in each episode, whereas the 'Episode Number' on the x-axis indicates the sequence of interactions the agent has had with its environment. For agent starting training from state 1, the average reward is 12.5629 for an episode (as shown in Figure 14) which consists of 20 slots, therefore the average reward per slot = 12.5629/20 = 0.6282.

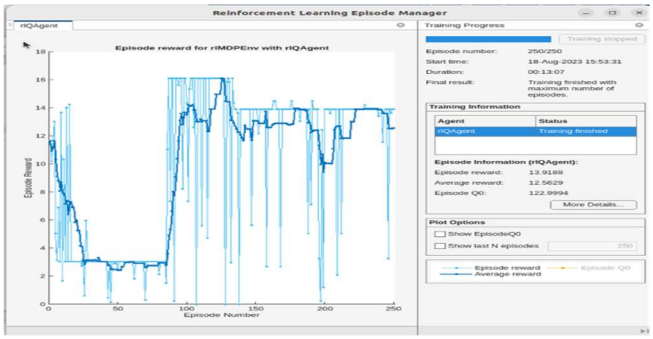

Fig. 14. RL Episode Manager at Episode 250

The Episode Manager of The RLib tool is shown in Figure 15 and includes additional metrics such as Actor Loss, Critic Loss, Entropy Loss, Loss and Score (i.e. Average Reward).

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/WCNC57260.2024.10570617, 2024 IEEE Wireless Communications and Networking Conference (WCNC)

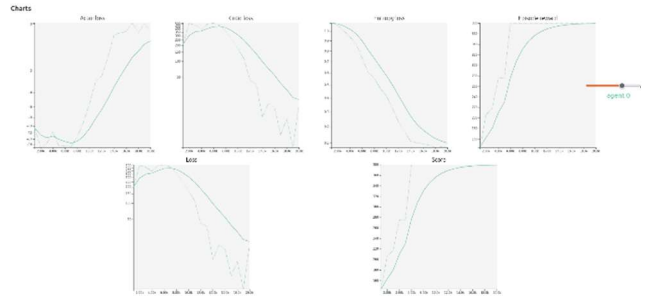

Fig. 15. RL Episode Manager at Episode 250

B. *Result overview for up to 2 state transitioning*

Up to two state transitioning between states restricts the transitions between states to [up, down, same, up2, down2]. The average is 12.6564 for an episode (as shown in Figure 16) which consists of 20 slots, therefore the average reward per slot = 12.6564/20 = 0.6328, which is again an efficiency between MCS/CQI 4 and 5 and marginally better than for the 1 state transitioning. It required around 2000 episodes for the RL state machine to reach a stable optimal state, which corresponds to 2000 5G frames of 10ms duration or 20 seconds in real time. This is considerably larger time required than for the single state transitioning system.

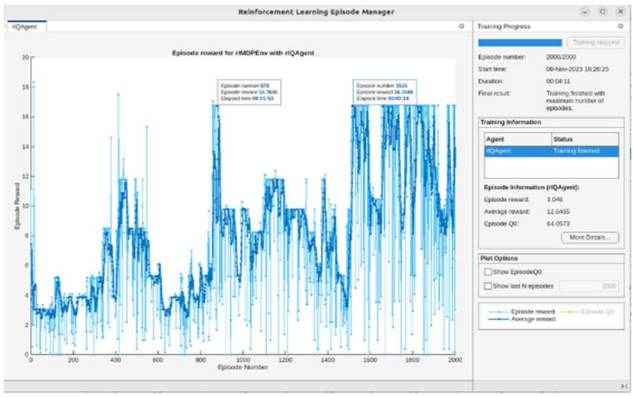

Fig. 16. Up to two State Transitioning Episode 2000

C. *Result overview for starting at best state*

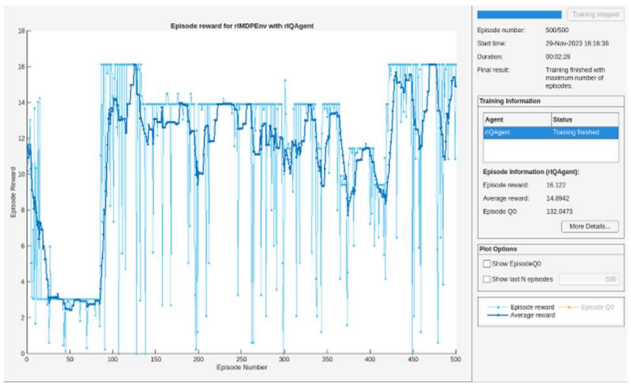

Fig. 17. Episode reward for rlMDPEnv with rlQAgent

Now for an agent starting training from the best state identified so far, the average reward is 14.8942 for an episode, as shown in Figure 17, which consists of 20 slots, therefore the average reward per slot = 14.8942/20 = 0.7447. This is an improvement from the 1 state solution starting from state 1.

VII. CONCLUSION

Workflow for training and creating MDP reinforcement learning agents for 5G RLC using MATLAB's Reinforcement toolbox and then use the Thales TheRLib MAD-DRL Tool to produce many light weight Python instances of these agents so that they can be simultaneously applied to multiple UE instances in a 5G CFN model.

The authors believe that this workflow can be applied to other autonomous network management such as slice management. It can also form the basis of the FlexRIC interface to ORAN.

ACKNOWLEDGMENT

The authors gratefully acknowledge support of EU Horizon 2020 Research Project 6G BRAINS (Bringing Reinforcement learning Into Radio Light Network for Massive Connections).

REFERENCES

[1] Marco Araújo et al "Final integration for AI-based E2E network slicing control and MANO" 6G BRAINS Deliverable D5.3, 30th September 2023.

[2] Techplayon, 2020. 5G NR Modulation and Coding Scheme – Modulation and Code Rate. [Online] Available at: https://www.techplayon.com/5g-nr-modulation-and-coding-scheme-modulation-and-code-rate/ [Accessed 27 July 2023].

[3] Santos, E.C., 2017. A simple reinforcement learning mechanism for resource allocation in lte-a networks with markov decision process and q-learning. arXiv preprint arXiv:1709.09312.

[4] K. A. Prof. John Cosmas, "6GBRAINS-T5.4 Brunel Powerpoint report," 2023

[5] MATLAB, 2023. Train Reinforcement Learning Agent in MDP Environment. [Online] Available at: https://uk.mathworks.com/help/reinforcement-learning/ug/train-reinforcement-learning-agent-in-mdp-environment.html?s_tid=srchtitle_site_search_2_MDP [Accessed 26 July 2023].

[6] Available at: https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da [Accessed 27 July 2023].

[7] Tang, L., Tan, Q.I., Shi, Y., Wang, C. and Chen, Q., 2018. Adaptive virtual resource allocation in 5G network slicing using constrained Markov decision process. IEEE Access, 6, pp.61184-61195.

[8] Wilhelmi, F., Bellalta, B., Cano, C. and Jonsson, A., 2017, October. Implications of decentralized Q-learning resource allocation in wireless networks. In 2017 ieee 28th annual international symposium on personal, indoor, and mobile radio communications (pimrc) (pp. 1-5). IEEE.

[9] "Train Reinforcement Learning Agent in MDP Environment" Available at: Train Reinforcement Learning Agent in MDP Environment - MATLAB & Simulink - MathWorks United Kingdom, seen 7/9/2023

[10] Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

[11] ETSI TS 138 214 V16.2.0 (2020-07) "5G; NR; Physical layer procedures for data" (3GPP TS 38.214 version 16.2.0 Release 16)

[12] Prince Kwaku Boakye "5G RAN Configuration and Control Using Reinforcement Learning" MSc Wireless Computer Communication Networks Dissertation Thesis, September 2023

[13] Alexandre Kazmierowski et al "Preliminary integration for AI-based E2E network slicing control and MANO" 6G BRAINS Deliverable D5.2, 21st December 2023

[14] Jobin Francis and Neelesh B. Mehta, "EESM-Based Link Adaptation in Point-to-Point and Multi-Cell OFDM Systems: Modelling and Analysis" IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 13, NO. 1, JANUARY 2014

[15] Josep Colom Ikuno, Martin Wrulich, Markus Rupp "System level simulation of LTE networks" On line Publication seen on 9-5-2021

[16] Raffaele Bruno, Antonino Masaracchia, Andrea Passarella "Robust Adaptive Modulation and Coding (AMC) Selection in LTE Systems using Reinforcement Learning" Ares(2015)1917862 - 06/05/2015.

[17] Kai Cui, Anam Tahir, Gizem Ekinci, Ahmed Elshamanhory, Yannick Eich, Mengguang Li and Heinz Koepp "A Survey on Large-Population Systems and Scalable Multi-Agent Reinforcement Learning" Online

Copyright © 2024 Institute of Electrical and Electronics Engineers (IEEE). Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. See: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/guidelines-and-policies/post-publication-policies/.