User Scheduling and Beam-Selection with Tabular and Deep Reinforcement Learning

Rebecca Aben-Athar, Cleverson Nahum, Davi da Silva Brilhante,
José F. de Rezende, Luciano Mendes and Aldebaro Klautau

***Abstract*— Reinforcement Learning (RL) is a promising alternative to traditional methods of user scheduling and beam-selection (SBS). Most of the current works on this topic adopt deep RL, in which neural networks allow to adopt state and action spaces with dimensions larger than the ones supported by tabular RL. However, while deep RL uses approximations that prevent them from getting policies that can be guaranteed to be optimal, tabular RL allows methods that find the optimal policy. The lack of optimal solutions complicates the proper interpretation and assessment of results in deep RL applied to SBS. This paper discusses how optimal policies can be found in the context of SBS and the associated issues. It also provides environments based on finite Markov decision processes that promote reproducible results and support a smooth transition from simple to more advanced RL problems. The presented experiments provide a benchmark of state-of-art deep and tabular RL algorithms, including scenarios for which the optimal solution is known. The results indicate that there is still room for improvement concerning deep RL algorithms, which do not reach the optimal solution in the adopted scenarios. This methodology not only provides insight into the performance of RL methods but helps compare new algorithms by first looking at contrived problems and later expanding the number of states and actions.**

***Keywords*— Reinforcement learning, optimal solutions, scheduling, beam-selection.**

I. Introduction

Millimeter Wave (mmWave) technology allows the 5th Generation (5G) networks and beyond to accommodate intense flows of data, providing high Quality of Service (QoS), higher packet throughput, and lower latency. As mmWave propagation is more prone to fading and blockage, it depends on massive Multiple Input Multiple Output (MIMO) techniques, such as *beamforming*, to produce directional beams. Analog beamforming allows directional beams that can increase range and improve data rate. In this scenario, *beam-selection* consists in searching for the best beam to achieve the best *combined channel* and better serve the scheduled user [1].

For efficient scheduling and beam-selection (SBS), the Base Station (BS) can leverage awareness of its surroundings, like the user locations [2] . Given the tools and concepts guiding the design of a Radio Access Network Intelligent Controller (RIC) in Open Radio Access Network (O-RAN) [3], [4], even extrinsic information, such as weather (rainy, sunny, etc.), can be incorporated into the SBS algorithm. Traditional schedulers are not able to easily incorporate such diverse sources of information. Reinforcement Learning (RL) is a good alternative in this case, given the flexibility it provides concerning the input information. This paper considers RL-based beam-selection and user scheduling.

In 5G and 6th Generation (6G) networks, decision processes, as the one performed in SBS, become increasingly complex and dynamic. For instance, one traditional way of gathering information about the environment relies on the transmission of pilot tones, but this leads to increased network overhead. Applications of RL aim at decreasing such overhead. The role of RL in mobile communications is becoming more relevant, since it is being used in resource allocation [5], network slicing [6], and congestion control [7], among others.

Some works do not use deep RL, but tabular. In [8], the authors used Q-learning, a tabular RL method, to perform beam tracking. In [9], multi-armed bandits were adopted to perform beam-selection using unmanned aerial vehicles (UAVs). But most recent work adopts deep RL. For instance, the authors in [10] used a deep Q-Network (DQN) to manage robots and improve the efficiency of mmWave MIMO systems. However, in this and other works, there is no discussion on how far from the optimal the presented results are.

The work in [11] is close to this paper in the sense that its authors also studied the optimality of the RL solutions by using a relatively small MIMO environment, for which the optimal solution can be obtained analytically. However, in the current paper, we also investigate scheduling, which brings extra difficulty due to the need of taking into account the buffers occupancy to avoid packet drops.

Our environments and simulation tools provide a benchmark of state-of-art deep and tabular RL algorithms, including scenarios for which the optimal solution is known. This methodology has as its main objective to compare the performance of optimal solutions and deep RL algorithms in simple SBS scenarios since it is usually unfeasible to provide optimal solutions in complex scenarios.

In summary, the contributions of this paper are:

- discussion of how to find optimal RL solutions in the context of SBS;
- comparison of state-of-art algorithms and benchmark definition;

Rebecca Aben-Athar, Cleverson Nahum and A. Klautau are with LASSE - 5G Group, Av Perimetral km 01, Guamá, 66075-750, Federal University of Pará, Belém, Brazil (e-mails: rebecca.athar@itec.ufpa.br, {cleversonahum, aldebaro}@ufpa.br). Davi da Silva Brilhante and José F. de Rezende are with Systems Engineering and Computer Science Program, Federal University of Rio de Janeiro, Rio de Janeiro-RJ 21941-914, Brazil (e-mails: {dbrilhante, rezende}@land.ufrj.br). Luciano Mendes is with the National Institute of Telecommunications (Inatel), Brazil (email: luciano@inatel.br). This work was partially funded by Brasil 6G project (RNP/MCTI grant 01245.010604/2020-14), SAMURAI project (FAPESP grant 20/05127-2) and CNPq-Brasil.

- open source software for running simulations and reproducing the results.

This paper is organized as follows: Section II presents the system model and describes RL-based SBS. Section III discusses the optimal solutions for the Finite Markov Decision Process (FMDP). Section IV presents the simulations results and Section V concludes the paper.

II. RL-Based Scheduling and Beam Selection

This section describes the adopted communication model and the RL systems.

A. Communication system model

This paper concerns downlink single-user massive MIMO systems in Vehicle-to-Infrastructure (V2I) scenarios. We assume a single cell with one BS and U User Equipments (UEs). The number of antennas at the BS is N_t and each UE has a single omnidirectional antenna. The BS is equipped with a Uniform Linear Array (ULA) and serves the UEs using an analog MIMO architecture with N_b beam vectors obtained from a Discrete Fourier Transform (DFT) codebook [1].

The incoming traffic is stored in buffers located at the BS. There is one buffer per UE, each with the capacity to store B packets. The UEs and the BS are positioned in a $G \times G$ grid-world, as depicted in Fig. 1. The convention adopted for the UE or BS position in this grid is $P(x, y)$ with $x \in \{0, 1, \ldots, G-1\}$ and $y \in \{0, 1, \ldots, G-1\}$. $P(0, 0)$ and $P(G-1, G-1)$ are the top-left and right-bottom corners, respectively. The UEs and BS cannot occupy the same position at the same time and the BS position is always $P(G-1, 0)$. At each time slot t, the UEs randomly move one position left, right, up, down, or stay at the same position.

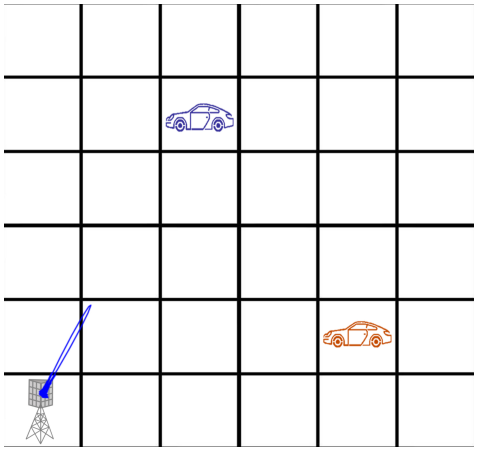

Fig. 1: Grid-world proposed for the SBS analysis using $G^2 = 36$ positions and $U = 2$ UEs.

This basic description can be used to derive several scenarios of interest, depending on the models adopted for the communication channel, traffic, and mobility. Table I illustrates some of the many options.

In this paper, when simulating mobility according to the ID *sm* in Table I, we assume the positions follow specific models of random walk in the two dimensions. This restriction does not allow one to take the UE direction into account but decreases the number of states that need to be taken into account.

TABLE I: Some possible SBS modeling assumptions.

Feature	ID
Channel	
Channel depends only on position and is kept constant over time, even along distinct RL episodes	fc
Still depends on position but channels vary over time, even within an episode, according to a stationary "small-scale" gain distribution.	vc
Traffic	
Full buffer: all users always have their buffers full of packets to be transmitted, and packets are never dropped	ft
Constant traffic: the same number of packets arrive at each time slot for all users	ct
Packet arrival is distributed according to stationary Poisson distributions (same statistics over time and episodes)	st
Packet arrival is distributed according to non-stationary Poisson distributions (statistics change at each episode and are kept constant within the episode)	nt
Mobility	
Number of UEs is kept constant over time, even along distinct episodes	fu
Number of UEs vary over time, even within an episode	vu
UEs move according to distributions that do not change over time nor episodes	sm
UEs move according to distributions that change over distinct episodes	nm

Without loss of generality, we assume that the spectral efficiency associated with the channel of a given user at time t, coincides with the number of packets that can be sent by this user at time t.

B. RL system

The RL agent is executed at the BS and has knowledge of the UE positions and their buffers occupancy. In the *scheduling* problem, the agent simply chooses one UE among U UEs. In the SBS problem, the agent also chooses one beam among the N_b beams to serve the UE that has been scheduled. In the following paragraphs, we present the details of the RL system.

The goal of the RL agent is to maximize the long-term return

$$R = \sum_{t=1}^{T_e} r_t, \tag{1}$$

defined as the accumulated reward r_t at time t over the episode duration $t = 1, \ldots, T_e$. The adopted reward aims at an agent minimizing the packet loss and is given by

$$r_t = -\ell_t, \tag{2}$$

where $\ell_t = \sum_{u,t} l_{u,t}, u = 1, \ldots, U$ is the sum of the number of packets lost by all users at time t.

With respect to the RL environment, this paper adopts the features *fc*, *ct*, *fu*, and *sm* in Table I. In this environment, we

assume the simplifying assumption that there are G^2 time-invariant channels that depend only on the UE's position (feature *fc*). The reason is to improve the readability of the results. We also fixed U for all time instants and episodes (feature *fu*), which means that the number of neurons in the output layer of a deep RL agent can be directly related to U.

The traffic model in the adopted environment is not a full buffer but constant (*ct*) and packets may be dropped. Hence, the scheduling requires an adequate definition of the state, which includes information about the buffers occupancy, and a focus on long-term return R to avoid an excessive number of dropped packets. An agent that can predict the channel spectral efficiency based on information about the UE trajectory can better schedule the users. This corresponds to a stochastic environment because the UEs mobility is not controlled by the agent, but these distributions are stationary.

We use the adopted environment for both the scheduling and SBS problems. For scheduling only, we will be able to obtain the optimal policies. For SBS only deep RL will be used because the computational cost is relatively large for the tabular RL methods.

1) Action: We assume that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment [12]. In this paper, the RL agent deals with SBS and does not control the users mobility. Hence, mobility is part of the environment and not of the agent's actions.

In the scheduling problem, the action would be the scheduling of one user among the U UEs, while SBS also takes into account the choice of one beam among the N_b beams to serve it. The action space dimension for SBS is

$$A = U \times N_b. \tag{3}$$

For instance, considering $U = 2$ UEs and $N_b = 4$ beam vectors, there are $A = 8$ actions in SBS and $A = 2$ in scheduling.

We model the RL *state* as the vector $\mathbf{s} = (\mathbf{p}, \mathbf{b})$, where $\mathbf{p}$ is the vector with the positions of all U users and $\mathbf{b}$ is the vector with their buffer occupancy. There are G^2 positions in the grid, but because one is occupied by the BS and the UEs cannot share the same position, the number of possible position permutations is $\prod_{i=1}^{U}(G^2 - i)$. Considering the buffer can be empty, the number of permutations for their occupancy is $(B+1)^U$. Therefore, the number S of states is

$$S = \left(\prod_{i=1}^{U}(G^2 - i)\right)(B+1)^U. \tag{4}$$

Assuming $G = 6$, $U = 2$ and $B = 3$, there are $S = 19,040$ states.

III. FMDP and Optimal Solutions

In this section, we first define notation and then discuss how to obtain optimal solutions for an FMDP.

A *generative* FMDP is a tuple $(\mathcal{S}, \mathcal{A}, p(s', r|s, a), \gamma)$, where γ is the discount factor, $\mathcal{S}$ and $\mathcal{A}$ are the finite sets of states and actions, respectively, and $p(s', r|s, a)$ if the joint probability for the next state s' and reward r given the current state s and action a (Eq. (3.2) in [12]). This model is called generative because the distribution $p(s', r|s, a)$ allows the creation of observations based on states and actions. In some scenarios, r does not depend on s', such that $p(r|s, a, s') = p(r|s, a)$. Another simplification that is often adopted in practice is that the reward is a deterministic function of the triple (s, a, s'). In this case, $p(r|s, a, s')$ is one for a single value of r and zero otherwise. If one is implementing the FMDP as a generative environment, the distribution $p(s', r|s, a)$ is needed and can be used, for instance, to randomly generate rewards according to $p(r|s, a, s')$ or to obtain the expected value $r(s, a, s')$ of the reward for any triple (s, a, s') as

$$r(s, a, s') = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}. \tag{5}$$

In many situations, one does not need a generative model and the goal is to *solve* the FMDP, which is also called using it *for control*. In this case, a *non-generative* FMDP model is more convenient,[1] and defined as a tuple $(\mathcal{S}, \mathcal{A}, p(s'|s, a), r(s', s, a), \gamma)$.

In this paper, solving the FMDP means to find the optimal *action-value function* $q_*(s, a)$ (defined in Eq. (3.16) of [12]). When the goal is simply to solve the FMDP, the distribution $p(s', r|s, a)$ is not needed, and it suffices to know $p(s'|s, a)$ and $r(s, a, s')$.

A. Tabular RL and Optimal Solutions

This subsection discusses optimal policies for the non-generative FMDP. Note that a policy is represented here as a matrix of dimension $S \times A$, providing a distribution over the possible actions for each state. A matrix with the optimal state values $q_*(s, a)$ can be easily converted into a policy [12]. Hence, the goal is to obtain $q_*(s, a)$. There are two issues to be circumvented: storage space and computational cost.

Regarding memory consumption, the representation of $p(s'|s, a)$ and $r(s, a, s')$ in software can rely on two distinct arrays, both with dimension $S \times A \times S$. In the adopted environment of $S = 19,040$ states with $A = 8$ SBS actions, each of these arrays demands approximately 10.8 GB considering elements represented with 4 bytes (single precision).

The computational cost is also $\mathcal{O}(S^2 A)$. When the Bellman equations are used to solve the FMDP (see, e. g., Eq. (3.16) of [12]), there are three nested loops, for state s, action a, and next state s'. The direct implementation of the algorithm may be highly inefficient when $p(s'|s, a)$ is sparse as in the adopted grid-world. For instance, when $S = 19,040$ states, finding $q_*(s, a)$ in a regular personal computer takes approximately 30 hours. But instead of an inner loop over s' ($S = 19,040$ in this example), one can pre-compute the feasible values of s' for each s and a. Using this approach, $q_*(s, a)$ can be found in approximately 15 minutes.

Despite this speedup of three orders of magnitude, one can note that FMDPs suffer not only from the $\mathcal{O}(S^2 A)$ scaling in

[1]The non-generative model is adopted, e. g., in the "RL Course" by David Silver, Lecture 2: Markov Decision Process, available at `https://www.youtube.com/watch?v=lfHX2hHRMVQ&t=2594s`.

storage and computational cost but from the fact that S often grows exponentially with the main environment parameters. To provide insight into alternative strategies to solve FMDPs, the next paragraphs discuss optimization via integer programming.

B. Integer Programming Optimal Solutions

For the scheduling-only version of the problem we also developed an integer optimization problem or Integer Programming (IP) problem. Such a problem is formulated to schedule the users while minimizing the packet loss and constrained by the spectral efficiency and buffer size limitations. Then, assume a set of U identical users moving on the grid as defined in Section II, with a constant packet demand of D_u packets and a buffer of size B.

The problem has three decision variables. The binary variable $s_{u,t}$ is equal to 1 if the user u is scheduled by the BS in the time-slot t. In each time-slot, the user that is scheduled sends a certain amount of packets, defined by $E_{u,t}$, which varies with the user u position in the time-slot t. If the D_u arriving packets are not all sent due to precarious channel conditions, the remaining packets are stored in the buffer and variable $q_{u,t}$ gives the buffer occupancy of user u at the time-slot t. Therefore, if the buffer is full, the arriving packets must be discarded. The variable $l_{u,t}$ is equal to the packets lost at the time-slot t by the user u. The whole IP formulation is described in Eq. (6).

$$
\begin{aligned}
P1: \min_{l} \quad & \sum_{u,t} l_{u,t} && \text{(6a)}\\
\text{s.t.} \quad & \sum_{u=1}^{U} s_{u,t} \leq 1, \ \forall\ t \in [0, N_e], && \text{(6b)}\\
& q_{u,t} \leq B, \ \forall\ u,\ t, && \text{(6c)}\\
& s_{u,t} \cdot E_{u,t} + q_{u,t} + l_{u,t} \geq q_{u,t-1} + D_u, && \text{(6d)}\\
& q_{u,t} \geq q_{u,t-1} - s_{u,t} \cdot E_{u,t}, && \text{(6e)}\\
& s_{u,t} \in \mathbb{B},\ l_{u,t}, q_{u,t} \in \mathbb{Z}^+, && \text{(6f)}\\
& u \in U, && \text{(6g)}\\
& t \in [0, N_e] && \text{(6h)}
\end{aligned}
$$

The objective function in Eq. (6a) minimizes the packet losses. As only one user can be scheduled per time-slot, constraint (6b) limits the sum of $s_{u,t}$ in u to be equal to 1 for each time-slot. Also, the maximum buffer size B upper limits the variable $q_{u,t}$, as in constraint (6c). The right-hand side of the constraint (6d) represents the sum of the packets already available in the buffer ($q_{u,t-1}$) and the newly arriving packets (D_u), for the user to take an action. Consequently, the left-hand side of constraint (6d) defines the three possible actions for the user u in time-slot t: to have the packets sent ($s_{u,t} \cdot E_{u,t}$), to put the packets in the buffer ($q_{u,t}$) or to discard some packets and put the arriving on the buffer ($l_{u,t}$). The constraint (6e) assures that if no packet is transmitted at t, the buffer occupancy is equal or greater to the occupation at $t-1$. Constraints (6f-6h) define the domains of the variables.

IV. Simulation Results

For evaluating models based on the adopted environment, 1000 episodes were created, each one with a duration $N_e = 1000$ time-slots. From this set, 800 episodes were reserved for training and 200 for validation and test. The packet arrival was kept constant at two packets per time-slot.

The grid is 6×6 ($G = 6$) positions, with the BS at position $(0, 5)$ and using a ULA with $N_t = 8$ antennas.

We assumed $U = 2$ users. The UEs mobility can be seen as non-uniform random-walk specified by distinct probabilities for each user. In case there is no restriction to any of the five possible movements of a UE, the first UE moves top or down with the probability of 0.3 and 0.4 is uniformly distributed to left, right and staying still. The second UE has similar behavior, but 0.3 is the probability of moving left or right, while 0.4 is uniformly distributed up, down, and staying still. In case any UE faces a movement restriction such as the end of the grid, that mass probability is uniformly reallocated to valid movements.

A. Results for scheduling-only

The scheduling problem consists of an FMDP with $S = 19,040$ states and $A = 2$ actions. It was tackled with three distinct approaches: a) solving it via the Bellman equations, b) solving it with IP, and c) using deep RL with distinct learning algorithms. These approaches will be identified as *Bellman*, *IP*, and by the name of the specific deep RL learning algorithm. While solving the Bellman equations iteratively for the scheduling problem required approximately 15 minutes, the IP problem took on average only 4.867 seconds.

Theoretically, both IP and Bellman solutions must provide the same results in this case. Therefore, we assumed that $p(s'|s, a)$ is not known by the agent and needs to be estimated by Monte Carlo [12]. This leads to discrepancies that impact the performance of the Bellman solution.

Due to its flexibility and easy implementation, we chose Stable Baselines 3 [13], to train the deep RL agents. Among the various RL libraries, we use 3 popular state-of-arts methods: PPO (Proximal Policy Optimization),combines ideas from A2C and use a trust region to improve actor, SAC (Soft Actor-Critic), which optimizes a stochastic policy in an off-policy way, and TD3 (Twin Delayed DDPG), which concurrently learns two Q-functions, by mean square Bellman error minimization. For all these algorithms we assumed the default values and did not tune their hyperparameters. The motivation for that was to evaluate how deep RL performs in such relatively easy problem and how the state-of-art algorithms compare with the optimal solution.

In Fig. 2 we show the results for the scheduling problem using histograms of all rewards obtained over the 200 test episodes. It can be seen that IP, which represents the optimal solution, reaches the best results, as expected. The Bellman solution is still better than all the three deep RL agents. This indicated that even in such simple scenarios, deep RL does not always get close to the optimal solution, and there is still room for improving deep RL. Having the optimal solutions allow a better understanding of what can be achieved.

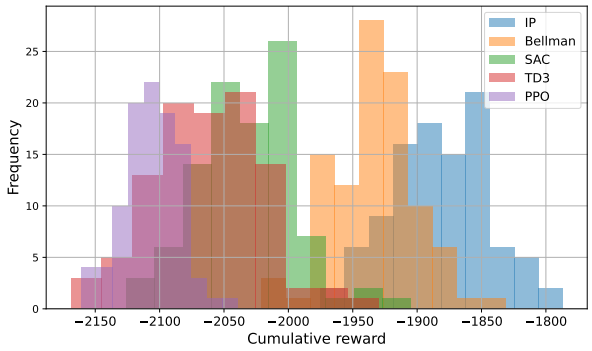

Fig. 2: Histogram of cumulative rewards for the scheduling problem using all test set.

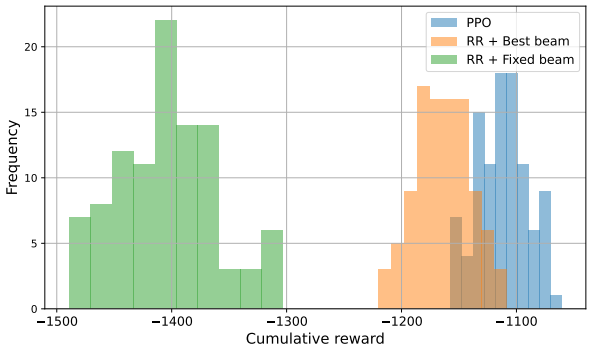

Fig. 3: Histogram of cumulative rewards for the scheduling and beam selection problem using all test set.

B. Results for scheduling and beam selection

We use a PPO RL method and round-robin (RR) variations to provide user scheduling and beam selection. Tabular and IP methods were not explored here due to the increased complexity of the scenario leading to a longer time to compute the optimum policies. Besides the user scheduling action for $U = 2$ users presented in Subsection IV-A, in this scenario, the agent also needs to define one beam among $N_b = 8$ beams to be used, where each beam has a specific spectral efficiency value in a given grid position. Therefore, the main agent goal is to select a combination of user and beam to minimize the number of dropped packets during the simulation.

The PPO RL agent considers an observation space containing the grid positions, spectral efficiencies, incoming packets, packets throughput, and buffer occupancy for each user. The agent's action space considers two discrete variables representing the index of the chosen user to be allocated and the beam index to be utilized. We developed two baselines: The first one utilizes a RR method for the user allocation combined with a fixed beam allocation that always uses the first beam available, and the second one combines a RR method to the user allocation with the best beam through exhaustive search.

Fig. 3 shows the histogram of cumulative rewards obtained to the PPO RL, RR using fixed beam, and RR using best beam selection over a specific episode. This result shows that the PPO agent outperformed all the baselines, even the RR using the best beam selection. It indicates that the PPO agent could learn an equivalent to optimal beam selection and a user scheduling process with better performance than the RR, confirming that PPO outperformed the baselines over all the test set.

V. Conclusion

The importance of deep RL in 5G and 6G is evident by the number of publications on the subject. However, there are several disadvantages when deep RL is used as a black box or investigated without connections to analytical results or optimal solutions.

The idea is that, despite their simplicity, these environments provide useful insight about how deep RL algorithms would perform in practical situations, indicating that there is still room for improvement concerning optimal solutions. Obtaining a near-optimal performance in a simple scenario increases the reliability of the deep RL performance in a complex scenario, since a complex scenario is usually unfeasible to provide an optimum solution to use for comparison.

References

[1] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 436–453, Apr. 2016.

[2] Y. Wang, A. Klautau, M. Ribero, A. C. K. Soong, and R. W. Heath, "Mmwave vehicular beam selection with situational awareness using machine learning," *IEEE Access*, vol. 7, pp. 87 479–87 493, 2019.

[3] B. Brik, K. Boutiba, and A. Ksentini, "Deep learning for B5G open radio access network: Evolution, survey, case studies, and challenges," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 228–250, 2022.

[4] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "Orchestran: Network automation through orchestrated intelligence in the open ran," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 270–279.

[5] X. Wang and T. Zhang, "Reinforcement learning based resource allocation for network slicing in 5G C-RAN," in *2019 Computing, Communications and IoT Applications (ComComAp)*, 2019, pp. 106–111.

[6] Y. Kim and H. Lim, "Multi-agent reinforcement learning-based resource management for end-to-end network slicing," *IEEE Access*, vol. 9, pp. 56 178–56 190, 2021.

[7] I. Nascimento, R. Souza, S. Lins, A. Silva, and A. Klautau, "Deep Reinforcement Learning Applied to Congestion Control in Fronthaul Networks," *Proceedings - 2019 IEEE Latin-American Conference on Communications, LATINCOM 2019*, pp. 1–6, 2019.

[8] H.-L. Chiang, K.-C. Chen, W. Rave, M. Khalili Marandi, and G. Fettweis, "Machine-learning beam tracking and weight optimization for mmwave multi-uav links," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5481–5494, 2021.

[9] S. Wasswa, M. S.Mojtaba, and G. Mohammad, "A fast machine learning for 5G beam selection for unmanned aerial vehicle applications," *Journal of Information Systems and Telecommunication (JIST)*, vol. 7, pp. 262–278, 2020.

[10] M. Feng and H. Xu, "Multi-robot enhanced intelligent multi-user millimeter-wave mimo systems under uncertain environment," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, 2019, pp. 965–969.

[11] H. Lee, M. Girnyk, and J. Jeong, "Deep reinforcement learning approach to MIMO precoding problem: Optimality and robustness," *CoRR*, vol. abs/2006.16646, 2020. [Online]. Available: https://arxiv.org/abs/2006.16646

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

[13] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, 2021.