Average Reward Reinforcement Learning for Wireless Radio Resource Management

Kun Yang*, Jing Yang†, and Cong Shen*
* Department of Electrical and Computer Engineering, University of Virginia, USA
† Department of Electrical Engineering, The Pennsylvania State University, USA

***Abstract*—In this paper, we address a crucial but often overlooked issue in applying reinforcement learning (RL) to radio resource management (RRM) in wireless communications: the mismatch between the *discounted* reward RL formulation and the *undiscounted* goal of wireless network optimization. To the best of our knowledge, we are the first to systematically investigate this discrepancy, starting with a discussion of the problem formulation followed by simulations that quantify the extent of the gap. To bridge this gap, we introduce the use of *average reward* RL, a method that aligns more closely with the long-term objectives of RRM. We propose a new method called the Average Reward Off-policy Soft Actor-Critic (ARO-SAC), which is an adaptation of the well-known Soft Actor-Critic algorithm in the average reward framework. This new method achieves significant performance improvement – our simulation results demonstrate a 15% gain in the system performance over the traditional discounted reward RL approach, underscoring the potential of average reward RL in enhancing the efficiency and effectiveness of wireless network optimization.**

***Index Terms*—Radio resource management, averaged reward reinforcement learning, deep reinforcement learning**

I. Introduction

In recent years, there has been a growing interest in applying reinforcement learning (RL) methods to solving radio resource management (RRM) problems in wireless networks. It largely stems from several important RL properties that match the characteristics of wireless networking. First, wireless network optimization is a closed-loop and sequential operation: set parameters, observe performance, and fine-tune. Second, many tasks in wireless networks have *long-term* performance impact, and their parameters are adjusted at a very low pace. As a result, the optimization cannot only target the immediate performance gain, but must take a long-term view. Third, there exist well-established feedback protocols in wireless standards, which provide a built-in mechanism for observing the state and receiving rewards. Lastly, RL research is a highly active and theoretically well-grounded area of machine learning, which lays a good foundation to its success in wireless networking.

Despite the promising initial results and the philosophical match, the majority of the existing solutions rely on the standard RL formulation, which maximizes *discounted* cumulative rewards in the long term. This objective, however, is misaligned with the typical objectives of wireless network optimization, where we do not treat future utility less importantly than the current one. A typical example is that we generally try to maximize the long-term average throughput of the entire network, treating both current and future user throughput equally in this formulation.

Naturally, one would ask whether we can design RL solutions for wireless network optimization that directly use undiscounted total reward as the objective. In the RL literature, this falls into the category of *average reward RL* [1]. To the best of the authors' knowledge, such average reward-based RL solutions have not been developed in wireless network optimization. In fact, the field of average reward RL itself is relatively under-explored. Only until recently have we seen the advancements to extend Policy Proximal Optimization (PPO) [2] and Deep Deterministic Policy Gradient (DDPG) [3] into the average reward framework. Nevertheless, these developments signal a growing potential for applying average reward RL in real-world engineering applications.

In this paper, we begin by pinpointing the discrepancy between the widely used discounted reward RL and the commonly adopted goals that are specific to RRM problems in wireless networks. Subsequently, we cast the RRM problem in an average reward RL framework. We develop a novel extension of the popular Soft Actor-Critic (SAC) algorithm to the average reward RL formulation, enhancing its applicability and effectiveness in addressing the RRM challenge. Our main contributions are summarized as follows.

1) To the best of our knowledge, we are the first to identify the discrepancy between the *discounted* reward RL formulation and the *undiscounted* objective of wireless network optimization. We showcase this discrepancy by re-formulating a RAN network slicing problem as an averaged reward RL one, and highlighting the mismatch of the design objectives of the prior RL approaches. We achieve this by unequivocally demonstrating the impact of the discount factor γ and environmental horizon on RAN slicing RRM in the prior solutions via numerical experiments.
2) We investigate how the practical algorithms handle the challenges of average reward rate estimation, and how the RL update is performed. Based on the estimation strategy for the off-policy RL algorithms introduced in ARO-DDPG [3], we extend the popular off-policy deep RL algorithm SAC to an average reward version

The work of K. Yang and C. Shen was partially supported by the U.S. National Science Foundation (NSF) under awards CNS-2002902, CNS-2003131, ECCS-2029978, ECCS-2030026, ECCS-2143559, and SII-2132700. The work of J. Yang was supported in part by the U.S. NSF under awards CNS-1956276, CNS-2003131 and CNS-2030026.

called **ARO-SAC** (Average Reward Off-policy SAC). With a tweak to the conventional TD error and Bellman equation, our new design enables SAC to perform with the average reward objective.

3) Our experimental result using an industry-grade wireless network simulator reveals that, with a properly selected hyperparameter, the proposed ARO-SAC can outperform the best SAC by a performance gain of 15%. We further investigate how the learning rates for the average reward rate and the environment horizon impact the performance of ARO-SAC.

The rest of the paper is organized as follows. The related works are surveyed in Section II. In Section III, we formulate the RRM problem using discounted reward RL and discuss the objective mismatch. We investigate the impact of discount factor and horizon in Section IV, which motivates us to develop the ARO-SAC algorithm in Section V. Section VI concludes the paper.

II. Related Works

RL for RRM: Due to its natural fit, RL-based solutions have been gradually adopted to solve RRM problems. Previous efforts include the solutions based on the bandit algorithms [4]–[8]. Subsequently, Q-learning-based algorithms were developed [9]–[11], followed by the adoption of the actor-critic architecture [12]–[16]. Regarding decentralized methods, multi-agent reinforcement learning (MARL) has solidified its relevance in [17]–[20]. More recent research has explored training RL policies using offline datasets [21], [22]. Despite these advancements, all methods predominantly rely on *discounted reward* RL algorithms, which overlooks a crucial aspect: the mismatch between the traditional objectives of wireless systems and the principles underlying discounted reward RL.

Averaged reward RL: Average reward RL, as a different formulation from the discounted reward RL setting, was designed to handle the scenarios where the future reward is of equal importance as the current one [1]. Most of the early works on average reward RL mainly focus on the tabular cases [2], [23], limiting their potential usage in complex environments. The initial development of average reward-based deep RL focused on Deep Q-Network (DQN) [24], which has limited performance compared to the actor-critic-based methods. The recent advancement in actor-critic-based average reward DRL algorithms [3], [25] has enabled the implementation of average reward RL for more practical problems.

III. Problem Formulation

In this section, we first establish the RRM problem within the context of RAN slicing. Then, we formulate the RRM problem into a discounted reward RL one. We then discuss the mismatch between the objectives of these two formulations.

A. RAN Slicing

In a RAN slicing system, we assume that the system has N slices in total, each handling a distinct user/traffic type. For these slices, our goal is to allocate packed radio resources properly to maximize the Quality of Service (QoS) of the whole system. These packed resources, named resource block groups (RBG), are then allocated to the users by the proportional fairness aware scheduler [26]. The system structure is illustrated in Fig. 1.

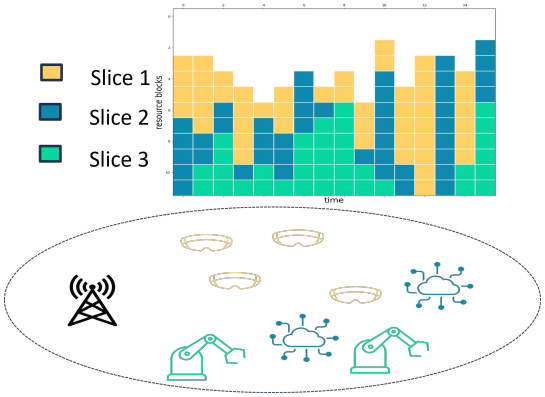

Fig. 1. Illustration of a RAN slicing system

Assume the system has M RBGs in total, and the QoS function at time t is $f_t(\mathbf{M(t)})$, where $\mathbf{M(t)} = [m_1(t), \cdots, m_N(t)]$, where $\mathbf{M(t)}$ is the resource allocation vector which stands for the resource blocks allocated to different slices and $m_i(t)$ stands for the resource allocated to slice i. Then, we can formulate the optimization problem as:

$$\begin{aligned} \underset{\mathbf{M(t)}}{\text{maximize}} \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} f_t(\mathbf{M(t)}) \\ \text{subject to} \quad & \sum_{i=1}^{N} m_i(t) \leq M. \end{aligned} \tag{1}$$

Clearly, the design goal of this formulation is to achieve the best possible *long-term average QoS*.

B. From RRM to Discounted Reward RL

In previous studies utilizing deep reinforcement learning (DRL) for the RRM problem in RAN slicing, the standard approach is to formulate the problem using discounted reward RL [**?**], [14], [15], [27]. In this section, we first discuss this discounted reward RL setting and then show where the mismatch happens.

As a concrete case of RAN slicing, we consider that in the optimization problem outlined in Eq. (1), two QoS metrics are pivotal: the total downlink throughput of the system and the average delay violation rate among users. The delay violation rate is defined as the proportion of packets exceeding the QoS latency threshold relative to the total number of packets a user receives. We then define the Markov Decision Process (MDP) for this RL problem as follows.

- **Observations:** Building on the considerations outlined above regarding system performance metrics such as throughput and delay violation rate, it is pivotal to monitor how much of the allocated resources have been utilized. Accordingly, we gather the following metrics for each slice in the system to serve as observations in our MDP: received traffic throughput T_{rx}, traffic load T_{tx}, resource utilization rate U, delay violation rate D_{vio}, and average one-way delay D_{avg} from every slice in the system. The observations are formally specified as

$$\{T_{\text{rx},i}, T_{\text{tx},i}, U_i, D_{\text{vio},i}, D_{\text{avg},i}\}_{i=1,\cdots,N}.$$

- **Actions:** As outlined in Section III-A, we need to allocate RBGs across different slices. Instead of distributing discrete resource units, our approach involves allocating a proportional share of RBGs to each slice, rendering our action a continuous variable within the range $[0, 1]$. Specifically, our action at time t is represented as $A(t) = [a_1(t), \cdots, a_{N-1}(t)]$, where each $a_i(t) \in [0, 1]$ denotes the proportion of RBGs allocated to slice i. We ensure the allocation is legitimate (i.e., $\sum_i a_i(t) \leq 1$) by integrating a softmax layer at the output of our policy network, ensuring a valid probability distribution over the slices.
- **Reward:** The reward design in a RAN slicing system should reflect its QoS objectives. Our configuration prioritizes two key components: the overall system throughput and delay violation rates. Accordingly, we construct our reward function as

$$R(t) = \sum_{i=1}^{N} r_i(t),$$

where each component of the reward, $r_i(t)$, is defined as:

$$r_i(t) = T_{\text{rx},i}(t) - \alpha D_{\text{vio},i}(t).$$

In our experiment, we set $\alpha = 4$ to impose a heavier penalty on the delay violations.

Assuming a discounted reward setting with the discount factor γ, the objective of this RL problem is:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(t)\right].$$

While this objective accumulates rewards over infinite time steps, *the influence of future rewards diminishes significantly due to the discount factor*. For instance, with $\gamma = 0.95$, rewards beyond 50 time steps contribute minimally to the objective, effectively accounting for only about 0.01 of their original value. This aspect of discounting does not align well with our initial goal as defined in Eq. (1), where the wireless network seeks optimal average performance over an infinite horizon. This mismatch motivates us to find better solutions to close the gap between the discounted RL and the original objective in our wireless network optimization problem.

TABLE I
EXPERIMENT PARAMETERS

Parameter	Value
Number of slices	3
Number of UEs per slice	$6 - 20$
Delay violation threshold	100 ms
Area	120×10 m^2
Downlink traffic	2 Mbp/s
Traffic pattern	Poisson arrival
UE mobility	$1 - 2$ m/s

C. Detailed Environment Setting

As described in Sec. III-A, we consider an RRM problem in a RAN slicing system with N slices and M RBGs. In our experiment, we have utilized **netgymenv** [28] as our simulator. We set $N = 3$ and $M = 25$. Our traffic model follows the LTE module in NS-3 [26]. To introduce different traffic flows for different slices, we assign a different user number to each of the slices ranging from 6 to 20. The detailed environment setting is given in Table I.

For the resource type we allocate to each of the slices, we utilize a setting similar to [**?**] where the soft slicing strategy is used. In a soft slicing system, when the resource is allocated to a slice, the users in this slice have priority in using these resources. The leftover resources can then be re-used by other users from different slices if the allocated resource is not fully used.

As for the RL algorithm, we use Soft Actor-Critic (SAC) [29] as our primary choice. We choose this algorithm mainly because we would like to see whether extending an existing deep RL algorithm to its average reward version is applicable.

IV. THE IMPACT OF DISCOUNT FACTOR AND HORIZON

We are not the first to identify the mismatch between the discounted reward RL and the real-world average return scenarios, where in [30], the authors have noticed that there exists a γ mismatch between the actors and critics. In [31], [32], the authors report supreme performance with large γ on long horizon tasks. In this section, we empirically establish that the same mismatch exists in the RRM problem for RAN slicing.

We conduct two key experiments to validate the mismatch between the discounted reward RL objective and the real wireless system goal. To verify the impact of horizon length, we incorporate a period reset signal, which resets the simulator after a predefined time step T. We regard this reset length as the period length of our environment. In the first experiment, we fix T and vary the discount factor γ, demonstrating that a larger γ improves performance by valuing the longer future more equally. In the second experiment, we fix a large γ and vary T, confirming that a larger γ can help the agents look into the longer future.

Fix T, vary γ: Table II illustrates that when T is constant, increasing γ consistently enhances the RL agent's performance. The result suggests that it is helpful in a system trying to maximize long-term average rewards to have a larger

discount factor, i.e. making the agent able to take longer steps into their consideration.

TABLE II
EXPERIMENTAL RESULTS WITH $T = 200$ AND DIFFERENT γ

γ	cumulative reward
0.9	10.53 ± 1.25
0.93	13.24 ± 0.52
0.95	14.20 ± 0.50
0.99	$\mathbf{15.67 \pm 0.37}$

Fix γ, vary T: When γ is fixed at a high value (e.g., 0.99), extending the horizon also results to an improved average reward per step, as evidenced by the results in Table III. This longer horizon also plays a pivotal role as it ensures the RL agent can learn the transition from a longer future.

TABLE III
EXPERIMENTAL RESULTS WITH $\gamma = 0.99$ AND DIFFERENT T

T	average reward
200	0.078 ± 0.002
500	0.079 ± 0.002
1000	0.082 ± 0.003
2000	$\mathbf{0.085 \pm 0.005}$

Summarizing these results, a heuristic solution emerges: set $\gamma = 1$ which would ensure that rewards do not decrease over time. However, in our experiment shown in Figure 2, naively setting γ to 1 appears beneficial for policy training initially but leads to significant instability later. This instability suggests that simply increasing γ is not optimal. Based on this observation, a new tool is needed to close the gap between discounted reward RL and the network optimization goal.

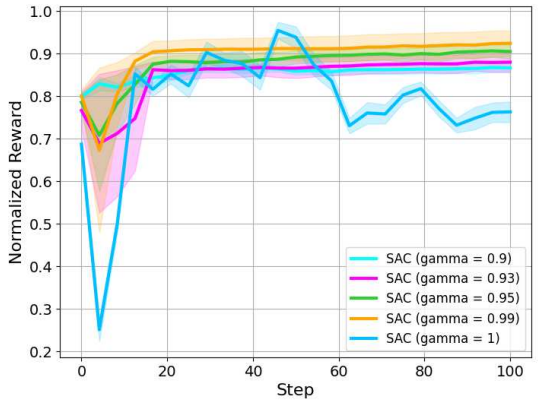

Fig. 2. Experimental results with $\gamma = 1$. Shadowed areas indicate the confidence intervals.

V. AVERAGE REWARD SOFT ACTOR-CRITIC

The concept of average reward RL, as the name suggests, is to maximize the long-term average reward for sequential decision-making problems [1]. In this section, we discuss the difference between the average reward RL and the discounted reward RL, and show how to change a discounted reward formulation into an average reward formulation. Then, following the design flow of ARO-DDPG [3], we extend the SAC principle to satisfy the average reward objective and conduct experiments to evaluate the performance of this new algorithm.

A. Re-formulation

The difference between the average reward RL and discounted reward RL primarily lies in their objective functions, where the average reward RL's objective is defined as:

$$\max_{\pi} \lim_{T \leftarrow \infty} \frac{1}{T} \sum_{t=1}^{T} r(t). \tag{2}$$

In Eq. (2), if we treat the reward function $r(t)$ as the QoS function in Eq. (1), the goal of the average reward RL exactly matches the wireless network objective.

To solve this new RL problem, the major workflow remains the same. From the principle in [1], similar to the discounted reward setting, we can solve the average reward RL through the Bellman equation. More specifically, we can still utilize the TD error-based method to solve the problem. The only difference is that, instead of having a discount factor γ, we now maintain an estimation of the true average reward ρ to help with the solution. The process of getting the average reward TD error is listed below.

In the average reward setting, we define the *differential* return, which measures the difference between the rewards and the true average reward, as

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots, \tag{3}$$

where π stands for the current policy. Based on this differential return, we then define the value function and compute the corresponding Bellman equation as

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \qquad (4)\\ &= \mathbb{E}_\pi[R_{t+1} - r(\pi) + G_{t+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r - r(\pi) \\ &\quad + \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)[r - r(\pi) + V_\pi(s')]. \qquad (5) \end{aligned}$$

From this Bellman equation and the definition of the TD difference, we can now compute the one-step TD as

$$\delta_t = R_{t+1} - \rho + V(S_{t+1}) - V(S_t). \tag{6}$$

For comparison, we note that the standard discounted reward TD error equals

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{7}$$

Comparing these two, we see that the new TD error in Eq. (6) substitutes the discount factor with the estimated average reward rate ρ.

B. Average Reward SAC

With the average reward TD error, if we have an accurate estimate of ρ, we can solve the RL problem by minimizing this error. However, this is a challenging task and two main methods have been adopted in average reward DRL. One is to collect the full trajectory of the policy and directly estimate ρ by setting

$$\hat{\rho} = (1-\alpha)\hat{\rho} + \frac{\alpha}{N}\sum_{n=1}^{N} r(s_n, a_n).$$

As described in [25], this type of estimation is more desirable for on-policy algorithms like PPO. The second choice is to make the average reward a trainable parameter and then to use gradient descent to update this parameter [3]. Mathematically we have

$$\hat{\rho}_{t+1} = \hat{\rho}_t + \nabla_\rho \varepsilon_t,$$

where

$$\varepsilon_t = r(s_t, a_t) - \rho_t - Q(s_t, a_t).$$

Since we use SAC as our primary discounted reward RL algorithm, which is an off-policy one, the latter choice is more applicable for designing the average reward SAC.

To develop SAC under an average reward setting, we take one step further from [3] and design ε_t for SAC as:

$$\varepsilon_t = r(s_t, a_t) - \rho_t - \min(Q_1(s_t, a_t), Q_2(s_t, a_t)). \quad (8)$$

Following this step, we extend the SAC algorithm into an average reward version and describe the complete procedure in Algorithm 1, where we mark the different steps incorporating ρ in the bold font.

C. Experiments

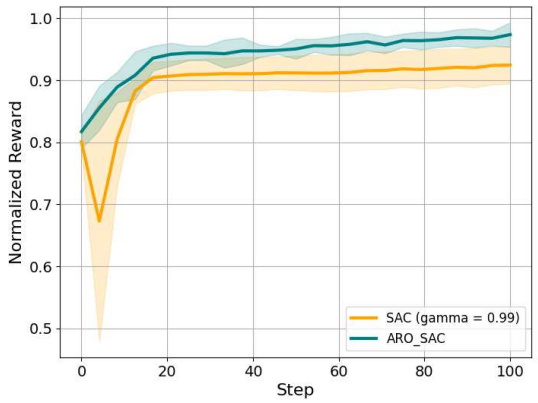

Fig. 3. Experimental result using ARO-SAC, where the experiment is averaged over 5 independent runs over 5 different combinations of user numbers.

In this section, we implement the proposed ARO-SAC in our simulation and compare its performance with the vanilla SAC with $\gamma = 0.99$. To verify the effectiveness of ARO-SAC, we compared the performance of the algorithm with pure SAC under horizon $T = 200$ and discount factor $\gamma = 0.99$.

Algorithm 1 Average Reward Off-Policy Soft Actor-Critic (ARO-SAC)

Initialize policy parameters θ, Q-function parameters ϕ_1, ϕ_2, **average reward estimator** ρ
Initialize target Q-function parameters $\phi_{targ,1} = \phi_1, \phi_{targ,2} = \phi_2$
Initialize environment and observe initial state s
Initialize replay buffer $\mathcal{D}$
for each time step **do**
    Sample action $a \sim \pi_\theta(\cdot|s)$ based on current policy
    Execute action a in the environment
    Observe reward r, new state s', and done signal d
    Store transition tuple (s, a, r, s', d) in replay buffer $\mathcal{D}$
    Sample random minibatch of transitions (s, a, r, s', d) from $\mathcal{D}$
    Compute target Q-value:
        $y = r - \rho + \min_{i=1,2} Q_{\phi_{targ,i}}(s', \tilde{a}')$
        where $\tilde{a}' \sim \pi_\theta(\cdot|s')$
    Update Q-functions by one step of gradient descent using:
        $\nabla_{\phi_i} \frac{1}{|B|} \sum (Q_{\phi_i}(s,a) - y)^2$ for $i = 1, 2$
    Update policy by one step of gradient ascent using:
        $\nabla_\theta \frac{1}{|B|} \sum \log \pi_\theta(a|s) Q_\phi(s,a)$
    Update average reward estimator ρ:
        $\nabla_\rho \frac{1}{|B|} \sum (\varepsilon_t)^2$
    Update target networks:
        $\phi_{targ,i} \leftarrow \tau\phi_i + (1-\tau)\phi_{targ,i}$ for $i = 1, 2$
    Observe new state $s \leftarrow s'$
end for

The result in Fig. 3 shows that the performance of our proposed ARO-SAC, while eliminating the drawback of an unstable convergence caused by setting $\gamma = 1$, also outperforms vanilla SAC with $\gamma = 0.99$ by 15%. This demonstrates a solid gain of utilizing average reward RL on this RRM problem in RAN slicing over the discounted counterpart. However, we also want to point out that while the average reward RL does help with the policy's performance, it introduces an extra trainable parameter that needs extra hyperparameter tuning (learning rate selection) steps. The learning rate for the parameter ρ needs careful selection. In our experiment, we set this learning rate to $1e-5$, which is slightly smaller than the learning rate of our actor-network.

VI. Conclusion

This paper addressed a critical mismatch between the conventional discounted reward reinforcement learning (RL) framework and the long-term objectives inherent to radio resource management (RRM) in wireless networks. We first validated this mismatch between discounted reward objectives and the actual goals of wireless systems. Our results underscored that even slight modifications toward considering longer-term outcomes, such as extending the horizon and adjusting the discount factor, could enhance performance under the discounted reward framework. We then developed

the Average Reward Off-policy Soft Actor-Critic (ARO-SAC), adapting the Soft Actor-Critic algorithm to the average reward framework, which significantly aligns with the long-term goals of RRM. Our experiments demonstrated a 15% improvement in the overall system performance over the conventional discounted reward RL approach, confirming the effectiveness and advantages of average reward RL in enhancing wireless network management. Interesting future works include providing theoretical guarantees of ARO-SAC and improving the algorithm design by reducing the hyperparameter fine-tuning.

REFERENCES

[1] A. G. Barto, "Reinforcement learning: An introduction by richards' sutton," *SIAM Rev*, vol. 6, no. 2, p. 423, 2021.

[2] S. Zhang, Y. Wan, R. S. Sutton, and S. Whiteson, "Average-reward off-policy policy evaluation with function approximation," in *international conference on machine learning*. PMLR, 2021, pp. 12 578–12 588.

[3] N. Saxena, S. Khastagir, S. Kolathaya, and S. Bhatnagar, "Off-policy average reward actor-critic with deterministic policy search," in *International Conference on Machine Learning*. PMLR, 2023, pp. 30 130–30 203.

[4] S. Nagaraja, F. Meshkati, M. Yavuz, S. Mitra, V. Khaitan, V. P. S. Makh, C. S. Patel, Y. Tokgoz, and C. Shen, "Power control for a network of access points," Nov. 2016, US Patent 9,497,714.

[5] N. Valliappan, C. Chevallier, A. D. Radulescu, and C. Shen, "Base station employing shared resources among antenna units," Sept. 2016, US Patent 9,451,466.

[6] Y. Huang, C. S. Patel, T. A. Kadous, M. Yavuz, L. Zhang, R. Prakash, V. Chande, C. Chevallier, S. Nagaraja, F. Meshkati *et al.*, "Methods and apparatus for power management in a wireless communication system," 2016, uS Patent 9,451,480.

[7] C. Shen, R. Zhou, C. Tekin, and M. van der Schaar, "Generalized global bandit and its application in cellular coverage optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 218–232, Feb. 2018.

[8] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1434–1446, Feb. 2019.

[9] K. I. Ahmed and E. Hossain, "A deep Q-learning method for downlink power allocation in multi-cell networks," *arXiv preprint arXiv:1904.13032*, 2019.

[10] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[11] G. Zhao, Y. Li, C. Xu, Z. Han, Y. Xing, and S. Yu, "Joint power control and channel allocation for interference mitigation based on reinforcement learning," *IEEE Access*, vol. 7, pp. 177 254–177 265, 2019.

[12] Y. S. Nasir and D. Guo, "Deep actor-critic learning for distributed power control in wireless mobile networks," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 398–402.

[13] K. Yang, C. Shen, and T. Liu, "Deep reinforcement learning based wireless network optimization: A comparative study," in *IEEE INFOCOM Workshop on Data Driven Intelligence for Networks*, Toronto, Canada, Jul. 2020, pp. 1248–1253.

[14] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, July 2022.

[15] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427–441.

[16] Y. S. Nasir and D. Guo, "Deep reinforcement learning for joint spectrum and power allocation in cellular networks," in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.

[17] ——, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[18] K. Yang, D. Li, C. Shen, J. Yang, S.-p. Yeh, and J. Sydir, "Multi-agent reinforcement learning for wireless user scheduling: Performance, scalablility, and generalization," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022, pp. 1169–1174.

[19] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3507–3523, 2021.

[20] Y. Zhang and D. Guo, "Distributed MARL for scheduling in conflict graphs," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2023, pp. 1–8.

[21] K. Yang, C. Shi, C. Shen, J. Yang, S. Yeh, and J. Sydir, "Offline reinforcement learning for wireless network optimization with mixture datasets," *IEEE Transactions on Wireless Communications*, vol. 23, no. 10, pp. 12 703–12 716, Oct. 2024.

[22] K. Yang, S.-P. Yeh, M. Zhang, J. Sydir, J. Yang, and C. Shen, "Advancing RAN slicing with offline reinforcement learning," in *2024 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2024, pp. 331–338.

[23] S. Zhang, B. Liu, and S. Whiteson, "Mean-variance policy iteration for risk-averse reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 905–10 913.

[24] O. Anschel, N. Baram, and N. Shimkin, "Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 176–185.

[25] X. Ma, X. Tang, L. Xia, J. Yang, and Q. Zhao, "Average-reward reinforcement learning with trust region methods," *arXiv preprint arXiv:2106.03442*, 2021.

[26] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[27] L. Geng, J. Dong, S. Bryant, K. Makhijani, A. Galis, X. de Foy, and S. Kuklinsk, "Network slicing architecture," Internet Engineering Task Force, Internet-Draft draft-geng-netslices-architecture-02, 2017, available online: https://datatracker.ietf.org/doc/html/draft-geng-netslices-architecture-02.

[28] M. Zhang and J. Zhu, "NetworkGym: Democratizing Network AI via Sim-aaS," https://intellabs.github.io/networkgym/, 2023.

[29] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[30] S. Zhang, R. Laroche, H. van Seijen, S. Whiteson, and R. T. d. Combes, "A deeper look at discounting mismatch in actor-critic algorithms," *arXiv preprint arXiv:2010.01069*, 2020.

[31] D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov, "Revisiting the minimalist approach to offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[32] J. Wu, H. Wu, Z. Qiu, J. Wang, and M. Long, "Supported policy optimization for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 278–31 291, 2022.