Decentralized Deep Reinforcement Learning Approach for Channel Access Optimization

Sheila C. da S. J. Cruz, Felipe A. P. de Figueiredo and Rausley A. A. de Souza

***Abstract*— The IEEE 802.11 standard's binary exponential back-off (BEB) algorithm for collision avoidance, although widely adopted, leads to sub-optimal network performance and significant bandwidth wastage, especially in dynamic and densely populated networks. Addressing these critical limitations, our research introduces a groundbreaking decentralized approach using advanced deep reinforcement learning (DRL) algorithms—specifically Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). By optimizing the contention window (CW) value, our method aims to not only maximize network throughput but also minimize collision rates. Through rigorous simulations with NS-3 and NS3-gym, our findings reveal that DQN and DDPG dramatically surpass BEB, delivering up to a 37.16% enhancement in network throughput in densely populated network scenarios. Moreover, our approach ensures consistent and robust throughput performance as the number of stations scales. These significant results underscore the transformative potential of DRL in revolutionizing wireless network efficiency and reliability, paving the way for more resilient and adaptive network management solutions in increasingly complex environments.**

***Keywords*— Wi-Fi, contention-based channel access, channel utilization optimization, reinforcement learning, NS-3, NS3-gym.**

I. Introduction

Wireless networks are widely used across various domains, requiring fair spectrum resource allocation to ensure optimal performance during data transmission. A key challenge is managing collisions, where simultaneous transmissions by multiple stations cause interference and data loss. The IEEE 802.11 standard uses the Carrier sense multiple access with collision avoidance (CSMA/CA) protocol in the Medium access control (MAC) layer to mitigate collisions by employing a CW value, a random back-off time that doubles with each new collision to minimize further occurrences [1]–[3].

The BEB algorithm manages this deferring method in CSMA/CA but has significant limitations, including sub-optimal results under high loads, lack of adaptability, and fairness issues [4], [5]. To address these drawbacks, machine learning-based solutions like DRL algorithms have been proposed. DRL algorithms can adapt to evolving network conditions, optimizing cumulative rewards over time and providing flexible, optimal solutions for various Wi-Fi network scenarios [6].

A centralized single-agent DRL approach, where the agent at the access point (AP) optimizes and broadcasts a unique CW value to all stations, has shown increased throughput [7]. However, a decentralized approach offers more robustness and efficiency, particularly in dense, dynamic scenarios. By leveraging distributed computing, new stations can be easily integrated, accelerating convergence to optimal collision avoidance solutions in wireless local area networks (WLANs) [8].

Several multi-agent reinforcement learning (MARL) methods have been proposed to enhance WLAN performance with decentralized solutions. For instance, one MARL approach optimizes spectrum occupation prediction and multi-channel slotted wireless network access, reducing inter-network collisions by 30% and increasing throughput by 10% compared to the traditional exponentially weighted moving average (EWMA) algorithm [9].

Another MARL algorithm for power allocation and joint subcarrier assignment in multi-cell orthogonal frequency-division multiplexing systems demonstrated 53.6% higher efficiency than conventional Q-learning. This approach, where each base station independently calculates resource allocation and collaborates for global optimization, is crucial for managing interference in heterogeneous networks with multiple APs and users sharing the same spectrum [10].

Previous works have limitations in optimizing network throughput and adaptability in extremely dense and dynamic scenarios, often showing high computational complexity [11]. Therefore, we propose a decentralized solution that treats each station as a DRL agent that updates its CW values, optimizing individual throughputs passed to the AP. The AP then broadcasts the total throughput value back to the stations, maximizing overall network performance and providing a more flexible, adaptable, and robust collision avoidance solution.

The proposed decentralized solution uses collision probability as the training metric for DRL agents. It compares the collision probabilities of the decentralized approach and the conventional BEB algorithm, showing that the decentralized method outperforms BEB and better adapts to changing network conditions, making it effective for collision avoidance in WLANs.

This work analyzes two scenarios: static (fixed number of nodes) and dynamic (increasing number of nodes). It proposes using DRL algorithms, specifically DQN and DDPG, with collision probability as the observation metric to optimize network performance.

The paper is structured as follows: Section II covers the theoretical background and methodology, Section III presents simulation results, and Section IV concludes with future work directions.

This work was partially funded by CNPq (Grant Nos. 403612/2020-9, 311470/2021-1, and 403827/2021-3), by Minas Gerais Research Foundation (FAPEMIG) (Grant Nos. APQ-00810-21, APQ-03162-24, and PPE-00124-23), by FADCOM - Fundo de Apoio ao Desenvolvimento das Comunicações, presidential decree no 264/10, November 26, 2010, Republic of Angola, and by the projects XGM-AFCCT-2024-2-5-1 and XGM-FCRH-2024-2-1-1 supported by xGMobile - EMBRAPII-Inatel Competence Center on 5G and 6G Networks, with financial resources from the PPI IoT/Manufatura 4.0 from MCTI grant number 052/2023, signed with EMBRAPII.

II. Simulation Methodology

A. Theoretical Background

Reinforcement learning (RL) involves a single agent learning through interaction with the environment to make decisions that maximize cumulative rewards [12]. The main goal of RL algorithms is to find a policy that effectively explores and exploits the environment to maximize total future rewards.

DRL combines RL with deep neural networks (DNNs) to handle high-dimensional data and accelerate convergence [13]. This study focuses on using DQN [14] and DDPG [15] algorithms to optimize CW, aiming to reduce node collisions and improve network performance.

A decentralized RL system is a multi-agent RL approach where multiple self-learning agents operate in a shared environment [16]. In MARL, agents independently maximize their rewards using local information without considering other agents, leading to a competitive learning process [17]. Decentralized MARL offers advantages such as optimal solutions for channel access, collision avoidance, resource management, and improved network performance, along with scalability and robustness. However, it faces challenges like non-stationarity, high computational complexity, and difficulty in achieving agent communication and collaboration due to limited local information and a lack of awareness of other agents' actions and rewards [16], [17].

B. Methodology

The proposed approach encompasses a decentralized algorithm that runs on multiple stations simultaneously. Each station observes the network state independently and selects appropriate CW values to optimize overall network performance. Next, we describe each part of the decentralized solution.

1) **Agent** is represented by a DRL algorithm (DQN or DDPG) to run on multiple stations varying from 5 to 50.
2) **Current state** is the environment status s, of all stations associated with the AP. However, it is impossible to get this information because of the nature of the optimization problem. Therefore, we model the problem as a partially observable Markov decision process (POMDP) instead of a Markov decision process (MDP). POMDP assumes the environment's state cannot be perfectly observed [18].
3) **Observation**, O, is the network information based on the collision probability to observe the overall network's status. This information is saved into a buffer of recent observations. Then, a moving average is calculated, producing the mean value, μ, and the variance, σ^2, transferred to a two-dimensional vector to train the DRL agent.
4) **Action**, a, determines the CW value. As we compare DRL algorithms with discrete and continuous action spaces, the actions are integer values between 0 and 6 in the discrete case and real values within the interval $[0, 6]$ in the continuous case. This interval is selected so that the action space is within 802.11 standard's CW range, which ranges from 15 up to 1023. Therefore, the CW value for each station can be calculated by applying $CW = \lfloor 2^{a+4} \rfloor - 1$.
5) **Reward**, r, is the normalized network throughput. It is calculated by dividing the actual throughput by the expected maximum throughput for each station. Each station's agent receives individual rewards, and the cumulative reward to be broadcast to every station is the sum of these individual rewards, resulting in a real value within the interval $[0, 1]$.

The collision probability is a good characterization of the environment state. It is the probability of collision, p_{col}, observed by the network. It can also be interpreted as the probability of transmission failure. It is calculated based on the number of transmitted, N_{t}, and correctly received, N_{r}, frames, that is

$$p_{\text{col}} = \frac{N_{\text{t}} - N_{\text{r}}}{N_{\text{t}}}. \tag{1}$$

The collision rate, which approximates the actual collision probability, becomes more accurate as the number of frames used for its calculation increases. This rate indicates the likelihood of a frame not being received due to simultaneous transmissions from other stations. Calculated during interaction periods, these probabilities provide insights into the performance of the selected CW value.

The proposed solution involves three stages: pre-learning, learning, and operational. Initially, parameters such as the number of stations, observation buffers, agent weights, and interaction periods are defined. During the pre-learning stage, each station collects data on transmitted and received frames, calculating observations and filling the observation buffer. The solution preprocesses this buffer to calculate a moving average of recent observations, generating mean (μ) and variance (σ^2) values for training the agent. In the learning stage, indicated by the current time exceeding a set update interval and the training flag being active, the agent selects CW values based on these statistics, and the DNNs are updated using samples from the replay buffer [15], [19]. Exploration of the environment is facilitated by adding a decreasing noisy factor to actions; in DQN, this corresponds to the probability of taking a random action instead of an action predicted by the agent, while in DDPG, Gaussian noise is added to actions. Still, in the learning state, throughput is computed, normalized to form a reward, and broadcast to all stations. This reward, along with the action and state information, is stored in the experience replay buffer, and agent weights are updated using a mini-batch from this buffer [15], [19]. The loss functions used to train the DQN and DDPG agents are detailed in [15], [19]. The entire learning process iterates through multiple episodes and steps, continually refining the CW optimization (i.e., updating the agents) for each station. The operational stage begins once the training period elapses, utilizing the learned actions without further training. The implementation of this proposed decentralized solution follows the pseudo-code shown in Algorithm 1, detailing the operation of these three stages.

Algorithm 1 DRL-based Decentralized CW Optimization

▷ ### Initialization ###
Define the maximum number of stations, WifiNode.
Initialize the observation buffer of each station, $O(i)$, with zeroes.
Initialize the weights of each agent, $\theta(i)$.
Get the action function for each station, $A_\theta(i)$, which each agent uses to choose the action according to its current state
Initialize the algorithm's interaction period with the environment, $envStepTime$
Initialize the number episodes, $N_{episodes}$
Initialize the number of steps per episode, N_{spe}
Initialize the training stage period, $trainingPeriod$
Set *trainingFlag* $\leftarrow True$ to tell the algorithm is in the training stage
Initialize the experience replay buffer of each station, $E(i)$, with zeroes.
trainingStartTime $\leftarrow$ *currentTime*
lastUpdate $\leftarrow$ *currentTime*
Initialize the previous mean value of each station $\mu_{prev}(i) \leftarrow 0$
Initialize the previous variance value of each station $\sigma^2_{prev}(i) \leftarrow 0$
Set $CW(i) \leftarrow 15,\ \forall i$

for $e = 1, \ldots, N_{episodes}$ **do**
  Reset and run the environment, i.e., reset and run the NS-3 simulator
  for $t = 1, \ldots, N_{spe}$ **do**
    for $i = 1, \ldots,$ WifiNode **do**

▷ ### Pre-learning stage ###
      $N_t(i) \leftarrow$ get number of transmitted frames of the ith station
      $N_r(i) \leftarrow$ get number of received frames of the ith station
      $observation(i) \leftarrow \frac{N_t(i) - N_r(i)}{N_t(i)}$
      $O(i)$.append($observation(i)$)

      if $currentTime \geq lastUpdate + envStepTime$ **then**

▷ ### Learning and operational stages ###
        $\mu(i), \sigma^2(i) \leftarrow preprocess(O(i))$
        $a(i) \leftarrow A_{\theta(i)}(\mu(i), \sigma^2(i), trainingFlag)$
        $CW(i) \leftarrow 2^{a(i)+4} - 1$

        if $trainingFlag == True$ **then**
          $N_{RP}(i) \leftarrow$ get the number of received packets of the ith station.
          $tput(i) \leftarrow \frac{N_{RP(i)}}{envStepTime}$
          Send the throughput of each station to the access point.
          $r \leftarrow normalize(tput(i))$
          Broadcast the new reward, r, to all associated stations
          E(i).append(($\mu(i)$, $\sigma^2(i)$, *a(i)*, *r*, $\mu_{prev(i)}$, $\sigma^2_{prev(i)}$))
          $\mu_{prev(i)} \leftarrow \mu(i)$
          $\sigma^2_{prev(i)} \leftarrow \sigma^2(i)$
          $mb(i) \leftarrow$ get random mini-batch from *E(i)*
          Update $\theta(i)$ based on *mb(i)*
        end if

        lastUpdate $\leftarrow$ *currentTime*
      end if

▷ ### Makes the transition between learning and operational stages ###
      if $currentTime \geq trainingStartTime + trainingPeriod$ **then**
        $trainingFlag \leftarrow False$
      end if
    end for
  end for
end for

III. Results

This section compares the proposed solution's performance against the conventional BEB algorithm under static and dynamic scenarios.

A. Simulation Scenario Description

In the decentralized solution, the stations with RL agents follow a distributed topology with one AP, as shown in Fig. 1. Each station acts as an autonomous agent, adjusting its CW value and receiving individual rewards. The cumulative reward is the total sum of all individual rewards. The stations' arrangement occurs statically and dynamically. The NS3-gym and NS-3 (version 3.29) simulators are used to train the agent

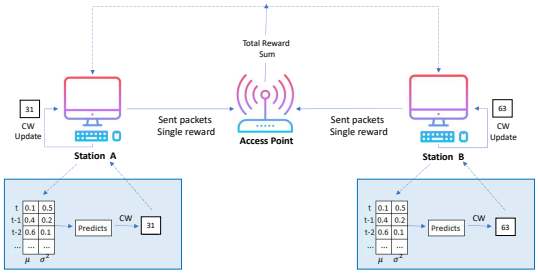

Fig. 1. System model for the decentralized DRL-based CW optimization.

TABLE I
NS-3 Environment configuration parameters.

Configuration Parameter	Value
Wi-Fi standard	IEEE 802.11ax
Number of APs	1
Number of static stations	5,15, 30 or 50
Number of dynamic stations	increases steadily from 5 to 50
Frame aggregation	disabled
Packet size	1500 [[bytes]
Max Queue Size	100 [packets]
Frequency	5 [GHZ]
Channel BW	20 [MHz]
Traffic	constant bit-rate UDP of 150 [Mbps]
MCS	HeMcs (1024-QAM with a 5/6 coding rate)
Guard Interval	800 [ns]
Propagation delay model	ConstantSpeedPropagationDelayModel
Propagation loss model	MatrixPropagationLossModel

and implement the DRL algorithm using TensorFlow (version 1.14.0) and PyTorch (version 0.4.1) [20]. The simulations were performed on a desktop with an Intel Xeon E5-1620 v3 processor, 32 GB RAM running Ubuntu 20.04.

Tables I and II summarize the parameters used in NS-3 for creating the agent's environment and NS3-gym for training the DRL agent, respectively. The agent's parameters, including learning and reward discount rates, were ideally chosen using a grid search from previous studies in the literature.

The DQN architecture features a recurrent long short-term memory (LSTM) layer with 8 cells, followed by two fully connected hidden layers containing 128 and 64 units, respectively. The architecture concludes with an output layer comprising 7 units. ReLU activation functions are applied in all layers except for the output layer, which utilizes a linear activation function. In contrast, the DDPG's Actor architecture includes an LSTM layer with 2 cells, a hidden layer with 32 units, and an output layer with a single unit. The DDPG's Critic architecture is similarly structured, featuring an LSTM layer with 2 cells, a hidden layer with 64 units, and an output layer with 1 unit. ReLU activation functions are employed in all layers of both the Actor and Critic models, except for the output layer, which again uses a linear activation function. Both DQN and DDPG leverage the Adam optimizer for training.

B. Static Scenario

In the static scenario, a fixed number of stations associated with the AP is kept constant during the experiment execution. As shown in Figure 2, the comparison of network throughput as a function of the number of stations highlights the superior performance of the decentralized DRL algorithms compared to

TABLE II

NS3-GYM AGENT CONFIGURATION PARAMETERS.

Configuration Parameter	Value
DQN's learning rate	4×10^{-4}
DDPG's actor learning rate	4×10^{-4}
DDPG's critic learning rate	4×10^{-3}
Reward discount rate	0.7
Batch size	32 samples
Replay memory size	18000 samples
Size of observation history memory	300 samples
$trainingPeriod$	840 [s]
$envStepTime$ (i.e., interaction interval)	10 [ms]

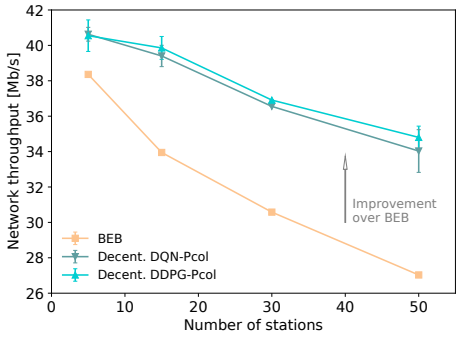

Fig. 2. Network throughput in the static scenario for 5 to 50 stations.

the conventional BEB algorithm. The improvements of DDPG over BEB are 5.19%, 17.64%, 16.67%, and 27.78% for 5, 15, 30, and 50 stations, respectively. The improvements of DQN over BEB are 5.19%, 17.21%, 16.27%, and 27.10% for 5, 15, 30, and 50 stations, respectively. The results demonstrate that DDPG is slightly better than DQN, especially for 15, 30, and 50 stations. This improved performance can be attributed to DDPG's ability to select any real CW value within the $[0, 6]$ range satisfactorily suitable for tracking the network's dynamics [15].

As illustrated in Figure 3, the mean CW value is shown across 15 episodes with 30 stations in the static scenario. The mean CW value is obtained individually for each station, Here we present the mean CW value corresponding uniquely to the 30$^{\text{th}}$ station. Furthermore, the results demonstrate that there is a higher variance on CW in the initial episodes, but as the training progresses, the variance reduces. This happens because the number of random actions decreases over time, and the agent converges to a result and learns correctly. After the 10$^{\text{th}}$ episode, the mean CW value is kept stable around the same value.

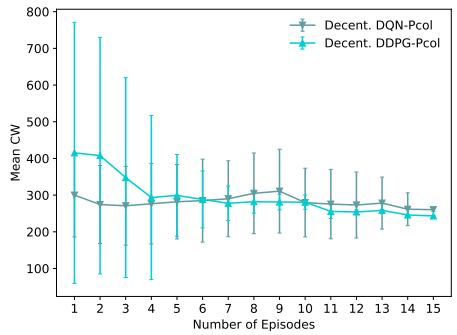

Fig. 3. Mean CW value for 30 stations in the static scenario.

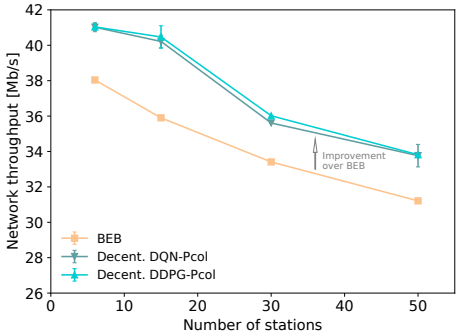

Fig. 4. Network throughput in the dynamic scenario for 5 to 50 stations.

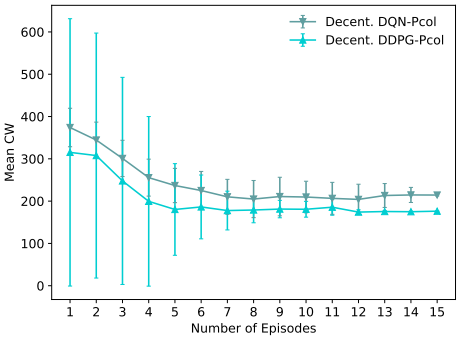

Fig. 5. Mean CW value for 30 stations in the dynamic scenario.

C. Dynamic Scenario

In this scenario, the number of stations grows steadily during the simulation execution, increasing from 5 to 50. The higher the number of stations, the higher the collision probability. This experiment evaluated whether the DRL algorithms correctly act upon network changes. As presented in Figure 4, it shows that the DRL algorithms effectively enhance the network's throughput compared to the BEB algorithm in the decentralized dynamic scenario. The degree of improvement in DQN and DDPG compared to BEB was similar. The improvements are 7.89%, 11.94%, 9.27%, 8.43% over BEB for 5, 15, 30, and 50 stations, respectively.

As shown in Figure 5, the mean CW value for the dynamic scenario is depicted as a function of the number of episodes. As with the static scenario, the CW value remains stable around the same value after some episodes with these mean CW values being exclusively related to the 30$^{\text{th}}$ station.

Figure 6 illustrates the instantaneous mean Contention Window (CW) calculated over 50 stations and the number of stations as a function of simulation time for the decentralized dynamic scenario, where the number of stations incrementally increases from 5 to 50 every 1.2 seconds. As the number of stations increases, the CW values are adjusted accordingly. It is observable that DQN varies between discrete neighboring CW values, while the DDPG consistently raises the CW value. This results in a lower CW for 50 stations, subsequently improving the throughput.

Figure 7 compares the instantaneous network throughput in the decentralized dynamic scenario, where the number of stations progressively increases from 5 to 50. The elevated number of stations modifies the CW value, affecting the instantaneous network throughput. The BEB's throughput

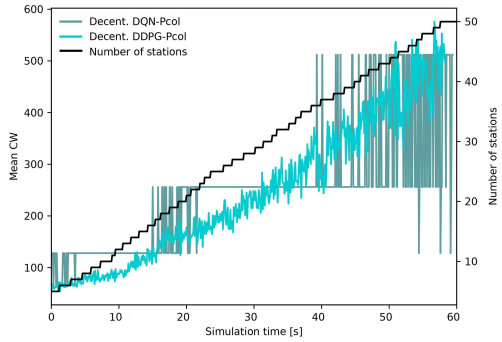

Fig. 6. Mean CW for 5 to 50 stations in the dynamic scenario.

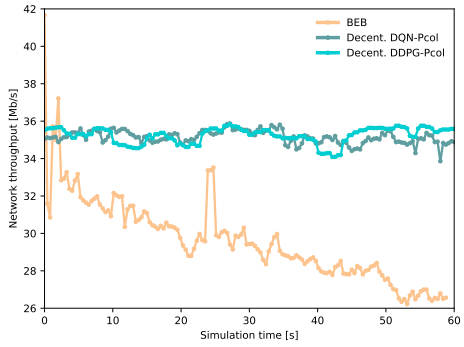

Fig. 7. Instantaneous network throughput in the dynamic scenario.

decays to approximately 26 Mbps when the number of stations connected with the AP reaches 50, unlike the decentralized mode, which yields a throughput of 35.8 Mbps with an increase of 37.16% over BEB. The proposed DRL algorithms (with either DQN or DDPG) present an almost constant behavior, keeping a high and stable throughput as the number of stations progressively increases. These findings make the DRL algorithm apt to address the collision avoidance challenges in dense wireless networks.

IV. Conclusions

This work has introduced a transformative decentralized approach utilizing DRL algorithms, specifically DQN and DDPG, to optimize the CW parameter and significantly enhance network throughput. The simulation results underscore the superiority of our approach over the traditional BEB algorithm, achieving an impressive up to 37.16% increase in network throughput with 50 stations. Both DQN and DDPG algorithms have demonstrated robust performance, proving their suitability for both static and dynamic network environments.

These findings highlight the potential of DRL algorithms to revolutionize collision avoidance strategies in WLANs. The enhanced throughput and stable performance across varying numbers of stations underscore the practical applicability and resilience of our approach in real-world scenarios.

Looking ahead, future work could explore the integration of station cooperation through information sharing, enabling agents to optimize network-wide rewards more effectively. Additionally, investigating the application of our DRL-based decentralized approach in other domains, such as IoT networks and vehicular ad-hoc networks, could further validate and extend the impact of our findings. The continued development and refinement of DRL algorithms for network optimization promise to drive significant advancements in wireless communication technologies, paving the way for more efficient, adaptable, and reliable networks.

References

[1] S. Giannoulis, C. Donato, R. Mennes, F. A. P. de Figueiredo, I. Jabandžic, Y. De Bock, M. Camelo, J. Struye, P. Maddala, M. Mehari, A. Shahid, D. Stojadinovic, M. Claeys, F. Mahfoudhi, W. Liu, I. Seskar, S. Latre, and I. Moerman, "Dynamic and collaborative spectrum sharing: The scatter approach," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019, pp. 1–6.

[2] E. C. Vilas Boas, J. D. e Silva, F. A. de Figueiredo, L. L. Mendes, and R. A. de Souza, "Artificial intelligence for channel estimation in multicarrier systems for b5g/6g communications: a survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, p. 116, 2022.

[3] X. Guo, S. Wang, H. Zhou, J. Xu, Y. Ling, and J. Cui, "Performance evaluation of the networks with Wi-Fi based TDMA coexisting with CSMA/CA," *Wireless Personal Communications*, vol. 114, no. 2, pp. 1763–1783, 2020.

[4] P. Patel and D. K. Lobiyal, "A simple but effective collision and error aware adaptive back-off mechanism to improve the performance of IEEE 802.11 DCF in error-prone environment," *Wireless Personal Communications*, vol. 83, pp. 1477–1518, 2015.

[5] B.-J. Kwak, N.-O. Song, and L. E. Miller, "Performance analysis of exponential backoff," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 343–355, 2005.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE communications surveys & tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[7] S. J. Sheila de Cássia, M. A. Ouameur, and F. A. P. de Figueiredo, "Reinforcement learning-based wi-fi contention window optimization," *Journal of Communication and Information Systems*, vol. 38, no. 1, pp. 128–143, 2023.

[8] A. Farhad and J.-Y. Pyun, "Resource management for massive internet of things in ieee 802.11 ah wlan: Potentials, current solutions, and open challenges," *Sensors*, vol. 22, no. 23, p. 9509, 2022.

[9] R. Mennes, F. A. P. De Figueiredo, and S. Latré, "Multi-agent deep learning for multi-channel access in slotted wireless networks," *IEEE Access*, vol. 8, pp. 95 032–95 045, 2020.

[10] Y. Hu, M. Chen, Z. Yang, M. Chen, and G. Jia, "Optimization of resource allocation in multi-cell OFDM systems: a distributed reinforcement learning approach," in *IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6.

[11] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. USA: Prentice Hall Press, 2009.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[17] D. Huh and P. Mohapatra, "Multi-agent reinforcement learning: A comprehensive survey," *arXiv preprint arXiv:2312.10256*, 2023.

[18] A. R. Cassandra, "A survey of POMDP applications," in *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, vol. 1724, 1998.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[20] P. Gawłowicz and A. Zubow, "NS3-gym: Extending openai gym for networking research," *arXiv preprint arXiv:1810.03943*, 2018.