



Article

Reinforcement Learning-based Wi-Fi Contention Window Optimization

Sheila C. da S. J. Cruz¹, Messaoud Ahmed Ouameur², and Felipe A. P. de Figueiredo^{*1}

¹ Instituto Nacional de Telecomunicações - INATEL, Santa Rita do Sapucaí, Brazil.
² Université du Québec à Trois-Rivières - UQTR, Québec, Canada.
^{*} Correspondence: felipe.figueiredo@inatel.br

Abstract: The collision avoidance mechanism adopted by the IEEE 802.11 standard is not optimal. The mechanism employs a binary exponential backoff (BEB) algorithm in the medium access control (MAC) layer. Such an algorithm increases the backoff interval whenever a collision is detected to minimize the probability of subsequent collisions. However, the expansion of the backoff interval causes degradation of the radio spectrum utilization (i.e., bandwidth wastage). That problem worsens when the network has to manage the channel access to a dense number of stations, leading to a dramatic decrease in network performance. Furthermore, a wrong backoff setting increases the probability of collisions such that the stations experience numerous collisions before achieving the optimal backoff value. Therefore, to mitigate bandwidth wastage and, consequently, maximize the network performance, this work proposes using reinforcement learning (RL) algorithms, namely Deep Q Learning (DQN) and Deep Deterministic Policy Gradient (DDPG), to tackle such a optimization problem. As for the simulations, the NS-3 network simulator is used along with a toolkit known as NS3-gym, which integrates a reinforcement-learning (RL) framework into NS-3. The results demonstrate that DQN and DDPG have much better performance than BEB for both static and dynamic scenarios, regardless of the number of stations. Moreover, the performance difference is amplified as the number of stations increases, with DQN and DDPG showing a 27% increase in throughput with 50 stations when compared to BEB. Furthermore, DQN and DDPG presented similar performances.

Keywords: Wi-Fi; contention-based access scheme; channel utilization optimization; machine learning; reinforcement learning; NS-3, NS3-gym

1. Introduction

The IEEE 802.11 or simply Wi-Fi is a set of wireless network standards designed and maintained by the Institute of Electrical and Electronics Engineers (IEEE) that defines MAC and physical layer (PHY) protocols for deploying wireless local area networks (WLANs). Their MAC layer implements a contention-based protocol, known as carrier-sensing multiple access with collision avoidance (CSMA/CA), for the nodes to access the wireless medium (i.e., the channel) efficiently [1,2]. With CSMA/CA, the nodes compete to access the channel as well as the radio resources [3,4].

One of the most critical parameters of the CSMA/CA mechanism is the contention window (CW) value, also known as back-off time, which is a random delay used for reducing the risk of collisions. If the medium is busy, an about-to-transmit Wi-Fi device selects a random number uniformly distributed within the interval $[0, CW]$ as its back-off value, which defers its transmission to a later time. CW doubles its value every time a collision occurs (e.g., when an ACK was not received), reducing the likelihood of multiple stations selecting the same back-off value. CW values range from the minimum contention window (CWMin) value, generally equal to 15 or 31 depending on the Wi-Fi standard, to the established maximum contention window (CWMax) value, which is equal to 1023. CW is reset to CWMin when an ACK is received, or the maximum number of re-transmissions

has been reached [5]. This deferring mechanism is also known as binary exponential back-off (BEB) [6].

In scenarios with few nodes, collisions will be less frequent and impactful, especially in static scenarios, where the number of nodes remains the same. On the other hand, in dynamic scenarios, where the number of nodes increases throughout time, collisions will be commonplace. Furthermore, the high number of collisions reduces the network throughput drastically since CW doubles its value when collisions are detected, leading to an inefficient network operation.

As can be seen, optimizing the CW value could be beneficial for Wi-Fi networks since the traditional BEB algorithm does not scale well when many nodes compete for the medium [7]. Once network devices with high computational capabilities become increasingly common, CW can be optimized through machine learning (ML) algorithms. The most common ML paradigms are supervised, unsupervised, and reinforcement learning. Supervised algorithms require a labeled dataset where the outcomes for the respective inputs are known. Still, creating such a dataset requires a model and a solution to the problem. Developing accurate models is, in several cases, a challenging and troublesome task. Besides that, many solutions are suboptimal, and supervised ML algorithms learning from datasets created with those solutions will never surpass its performance since the algorithm tries to replicate the input to label mapping present in the dataset [8].

Unsupervised learning occurs when the ML algorithm is trained using unlabeled data. The idea behind this paradigm is to find hidden patterns in the data. Finally, reinforcement learning occurs when the ML algorithm (a.k.a. agent in this context) interacts with the environment through trial and error action attempts without requiring labels, only reward information about the taken actions. This paradigm allows the agent to explore by taking random actions and finding optimal solutions. Therefore, since there is no optimal solution for the CW optimization (BEB is known to be suboptimal) [9–11], RL is the right choice for the learning paradigm.

This paper proposes using DQN and DDPG reinforcement algorithms to optimize the CW value and improve the performance of Wi-Fi networks by maintaining a stable throughput and minimizing collisions. DQN was chosen because it is relatively simple and has discrete action space. However, despite its simplicity, DQN generally displays performance and flexibility that rivals other methods [12]. DDPG was selected since it is a more complex method that represents actions as continuous values, yielding an exciting comparison with DQN [13]. We propose an RL-aided centralized CW optimization mechanism, which aims to maximize Wi-Fi networks' throughput by properly setting CW values. Therefore, the main contributions of this work are as follows:

1. A centralized mechanism for optimizing the CW in Wi-Fi networks using RL algorithms.
2. Comparison of DQN and DDPG RL algorithms with the traditional BEB.
3. A comparison between RL algorithms with discrete and continuous action spaces, namely, DQN and DDPG, respectively.
4. An optimization solution that applies to any of the 802.11 standards.

The remainder of the paper is organized as follows. Section II discusses related work. Section III presents a brief machine learning overview. Section IV describes the materials and methods used in the simulations. Section V presents the simulation results. Finally, Section VI presents conclusions and future works.

2. Related Work

The literature presents adequate and excellent contributions of machine learning (ML) methods applied to CW optimization in wireless networks. For example, in [14], the authors propose a CW optimization mechanism for IEEE 802.11ax under dynamically varying network conditions employing RL algorithms. The RL algorithms were implemented on the NS-3 [15] simulator using the NS3-gym [16] framework, which enables integration with python frameworks [16]. They proved to have efficiency close to optimal according

to the throughput result that remained stable even when the network topology changed dynamically.

To allow channel access and a fair share of the unlicensed spectrum between wireless nodes, the authors in [17] propose an intelligent ML solution based on CWmin (minimum CW) adaptation. The issue is that aggressive nodes, as they refer to in the paper, try to access the medium by forcefully choosing low CWmin values, while CSMA/CA-based nodes have a fixed CWmin set to 16, leading to an unfair share of the spectrum. The intelligent CW ML solution consists of a random forest, which is a supervised classification algorithm. Simulations were conducted on a C++ discrete-event-based simulator called CSIM [18] to evaluate the algorithm's performance. It was possible to obtain high throughput efficiency while maintaining fair allocation of the unlicensed channels with other coexisting nodes.

In [19], the authors present a Deep Q-learning algorithm to dynamically adapt CWmin to random access in wireless networks. The idea is to maximize a network utility function (i.e., a metric measuring the fair use of the medium) [20] under dynamic and uncertain scenarios by rewarding the actions that lead to high utilities (efficient resource usage). The proposed solution employs an intelligent node, called node 0, that implements the DQN algorithm to choose the CWmin for the next time step from historical observations. The simulation was conducted on NS-3 to evaluate the performance against the following baselines: optimal design, random forest classifier, fairness index, optimal constant, and standard protocol (with its CWmin fixed at 32). Two scenarios were considered for the simulation. The first scenario uses two states and follows a Markov process for CWmins of all nodes except node 0. The RL algorithm and random forest classifier reach outstanding performance for this case. The second scenario considers five states, and CWmins of all nodes different from node 0 follows a more complex process. The RL algorithm achieves utility close to optimal when compared to a supervised random forest classifier.

In [7], the authors propose an ML-based solution using a Fixed-Share algorithm to adjust the CW value to improve network performance dynamically. The algorithm comprises CW calculation, Loss/Gain function, and sharing weights. The algorithm considers the present and recent past conditions of the network. The NS-3 network simulator was used to evaluate the proposed solution and the performance metrics used were average throughput, average end-to-end delay, and channel access fairness. The Fixed-Share algorithm achieves excellent performance compared to the other two conventional algorithms, namely binary exponential backoff (BEB) and History-Based Adaptive Backoff (HBAB).

To optimize CW in a wireless local area network, [21] presents three algorithms based on genetic fuzzy-contention window optimization (GF-CWO), which is a combination of fuzzy logic controller and a genetic algorithm. The proposed algorithm is intended to solve issues related to success ratio, packet loss ratio, collision rate, fairness index, and energy consumption. Simulations were conducted in Matlab in order to evaluate the performance of the proposed solution, producing better results when compared to the BEB.

To avoid packet collisions in Mobile ad hoc networks (MANETs) [22], the authors of [23] propose a Q-learning-based solution to optimize the CW parameter in an IEEE 802.11 network. The proposed CW optimization method considers the number of packets sent and the collision generated from each station. Simulation results show that selecting a good CW value improves the packet delivery ratio, channel access fairness, throughput, and time. The benefits are even more significant when the queue size is less or equal to 20.

In [24], a different approach to control the CW value, named contention window threshold, is used. It employs deep reinforcement learning (DRL) principles to establish a threshold value and learn the optimal settings under various network scenarios. The method used is called the smart exponential-threshold-linear backoff algorithm with a deep Q-learning network (SETL-DQN). Its results demonstrate that this algorithm can reduce collisions and improve throughput.

The ML-based approaches presented in this section show how well ML algorithms can be applied to reach optimal performance in the wireless network field. Therefore, this

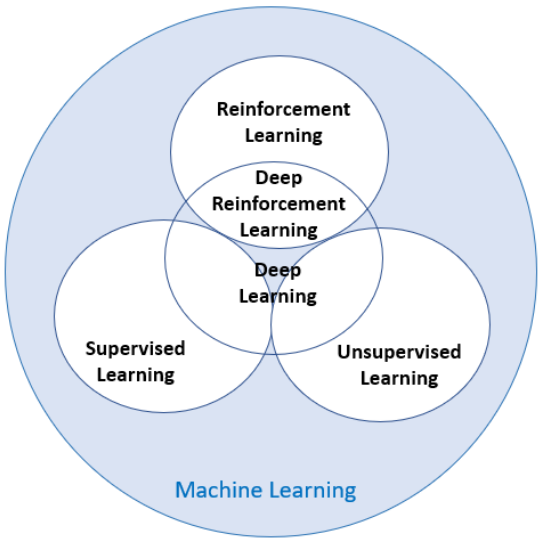


Figure 1. ML learning paradigms based on the type of training.

motivated us to study and propose an ML-based solution to reduce collisions by optimizing CW in different scenarios.

3. Machine Learning Overview

ML algorithms, also called models, have been widely applied to solve different problems related to optimization in wireless network communications systems [25–28]. These algorithms construct a model based on historical data, known as a training dataset, to perform tasks, for example, solving optimization problems, without being explicitly programmed to do so [29,30]. ML algorithms can provide self-management, self-learning, and self-optimizing solutions for an extensive range of issues in the context of dynamic resource allocation, spectrum resource management, wireless network optimization, and so much more [31].

The learning process of an ML model is called training, and it is used for the model to gain knowledge (i.e., infer a solution) and achieve the desired result. It is possible to classify the ML model learning based on the type of its training, also called learning paradigm [32]. The learning paradigms can be classified as: **Supervised learning**, **Unsupervised learning**, and **Reinforcement Learning (RL)**. Figure 1 shows the relations between the ML paradigms. Therefore, next, we provide a brief overview of these learning paradigms.

3.1. Supervised Learning

In this paradigm, the ML model uses labeled data during the training phase. Each input sample is accompanied by its desired output sample, called label. It is suitable for applications that have plenty of historical data [33]. Some well-known algorithms following this paradigm are linear and logistic regression, support vector machine (SVM), K-nearest neighbors (KNN), and artificial neural networks (ANN).

3.2. Unsupervised Learning

Here in this paradigm the ML model learns to find (sometimes hidden) useful patterns by exploring the input data without labels (i.e., without expected output values). The model is trained to create a small representation of the data [33]. This learning paradigm includes the following algorithms: K-means, isolation forest, hierarchical clustering and expectation–maximization.

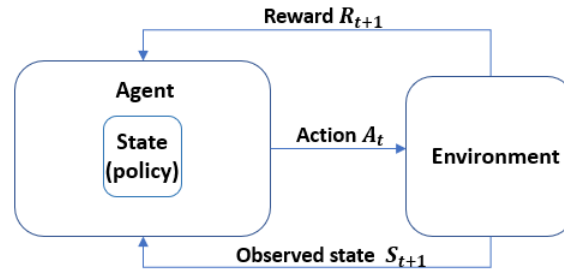


Figure 2. An RL agent interacting with the environment.

3.3. Reinforcement Learning

In this learning paradigm, an agent, in this context, the ML model, learns by continuously interacting with the environment and decides which action to take based on its own experience, mapping the current observed state of the environment to an action. The agent aims to learn a function, known as policy in this context, that models the environment and maps observed states into the best actions. The agent performs a decision-making task by trial and error in a self-learning manner [33]. This paradigm includes the following algorithms: Q-learning, Deep Q-learning, policy gradient learning, deep deterministic policy gradient, and the multi-armed bandit.

The environment is where the information (observed state and reward) is produced, and it has a dynamic nature compared to supervised and unsupervised learning paradigms. The Markov Decision Process (MDP) is generally adopted to represent the environment because it has a mathematical structure suitable for modeling decision-making problems [33]. It consists of a tuple of five elements $\mathbf{M} = \{S, A, P, \gamma, R\}$, where S is the state space, A is the action space, P is the transition probability, γ is the discount factor, and R is the reward. A reward is a positive or negative numeric value that indicates the quality of the action taken at a particular state [29]. The higher the reward, the better the action taken at that state. Conversely, the lower the reward, the worse the action. RL algorithms aim to find a policy that maximizes the total future reward. Figure 2 shows the interaction of the agent with the environment, which occurs in the following way: the agent observes (senses) the current state of the environment, S_t , based on this observation, the agent selects an action, A_t , and executes the action in the environment, this action on the environment returns information (i.e., results) in the form of reward, R_{t+1} and next-state, S_{t+1} , reached due to the action taken at the t -th time interval. Next, we explain how the policy is learned through a process of exploration-exploitation of the environment.

Policy: is a rule that helps the agent select the best action in a specific state. The primary objective of an RL algorithm is to learn a policy that maximizes the expected cumulative reward. The policy is a function that gives the probability of taking a given action when the environment is in a given state. The policy is learned by employing a method of exploring unknown actions in a given state and exploiting the current acquired knowledge. There must be trade-off between exploration of the environment and exploitation of the learned policy [34]. Simple exploration-exploitation methods are the most practical and used ones. One such method is the ϵ -greedy, where $\epsilon \in [0, 1]$ is a parameter controlling the amount of exploration and exploitation [35]. Normally, ϵ is a fixed hyper-parameter, but it can have its value decreased so that the agent explores the environment progressively less. Eq. 1 summarizes the ϵ -greedy exploration-exploitation mechanism used by RL algorithms to learn the best policy.

$$\text{Action} = \begin{cases} \text{Random action (Exploration),} & \text{if random number} < \epsilon, \\ \text{Best long-term action (Exploitation),} & \text{otherwise.} \end{cases} \quad (1)$$

Exploration: in this phase, the agent randomly selects an action from a uniformly distributed random variable with the number of possible values equal to the number of

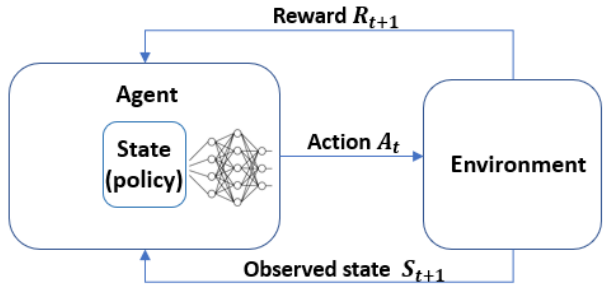


Figure 3. DRL Structure.

actions. Non-optimal actions are chosen to explore the environment, i.e., uncharted actions in a given state.

Exploitation: in this phase, the agent selects the action with the maximum quality value for that given state, i.e., it selects the action that has the best long-term effect in maximizing the expected cumulative reward.

3.4. Deep Reinforcement Learning

DRL is an improved extension of RL that integrates deep learning (DL), i.e., ANNs, with reinforcement learning algorithms [36]. This integration happens because RL algorithms present limitation problems related to space/action spaces, computational, and sample complexity [37]. That is, RL algorithms are not scalable and are limited to low-dimensional data issues, i.e., problems with small number of actions and states [38]. Therefore, the integration with DL improves the scalability issue and makes RL algorithms support high-dimensional data tasks. Compared to conventional RL, DRL explores a large dimensional neural network to speed up convergence. Figure 3 shows the deep reinforcement learning structure. The interaction with the environment occurs in the same way as with the RL agent. The only difference is that now the agent is an ANN model. DRL include the following algorithms: DQN, DDPG, Twin Delayed Deep Deterministic Policy Gradient (TD3), and Double Deep Q-learning Network (DDQN). This work focus on the study of using DQN and DDPG algorithms to assist in the optimization of CW with the primary goal of reducing node collisions while improving network performance. This way, next, we present a brief overview of these two DRL algorithms.

3.4.1. Deep Q Network

DQN is an off-policy DRL algorithm based on Q-learning with discrete action space, and continuous state space [39]. It is the result of incorporating deep learning into RL since, in many practical situations, the state space is high-dimensional and cannot be solved by traditional RL algorithms. Being an off-policy algorithm means that DQN uses an experience replay memory, where the agent learns from a batch of randomly selected prior experiences instead of the most recent one [40]. This random set of past experiences mitigates the bias that might stem from the fact that some environments have a sequential nature [40]. Here the agent is represented by a deep neural network that uses the state as the input of the neural network, and the output is the Q-value corresponding to a specific action.

The Q-value represents the quality of the action in a given state. The idea is for the neural network to output the highest Q-value for the action that maximizes the cumulative expected reward. However, using a single neural network renders training very unstable [36]. The trick to mitigate this problem and add stability to the training process is using two neural networks, predictive and target networks. They have the same structure (i.e., number of layers and neurons in each layer and activation functions) but have their weights updated at different times. The weights of the target network are not trained. Instead, they are periodically synchronized with the weights of the predictive network. The idea is that fixing the target Q values (outputs of the target network) for several updates will improve

the predictive network's training stability. DQN employs batch training and experience replay memory, making the agent learn from randomly sampled batch experiences. It also employs the ϵ -greedy exploration-exploitation mechanism.

3.4.2. Deep Deterministic Policy Gradient

DDPG is another off-policy DRL algorithm with continuous action space proposed in [41]. It is the result of the combination between deterministic policy gradient (DPG) and DQN algorithms, the former related to the actor-critic algorithm [42,43]. DQN avoids instability during the Q-function learning by employing a replay buffer and a target network. DQN has a discrete action space, while DDPG extends it to a continuous action space. The algorithm simultaneously learns a Q-function and a policy. Since DDPG inherits from the actor-critic algorithm, it is a combination of both policy (actor) and Q-value (critic) functions, where the actor takes actions according to a specific policy function, and the critic plays the role of an evaluator of the action taken [43]. The Q-value obtained by the critic indicates to the agent how good was the action for that policy.

DDPG consists of four networks: actor prediction, critic prediction, actor target, and critic target networks. The target networks have their weights copied from the prediction networks periodically. As with DQN, this procedure is adopted in order to stabilize the learning process, moving the unstable problem of learning the action-value function to a stable supervised learning problem [41].

Similarly to DQN, DDPG uses an experience replay memory to minimize correlations between samples. Regarding the policy aspect of exploration and exploitation, DDPG differs from DQN. Since DDPG works in continuous action space, exploring such space constitutes a significant problem. However, as it is an off-policy algorithm, the exploration problem can be treated independently from the learning algorithm [41]. DDPG creates an exploration policy that adds a noise value to the actor policy to solve this issue. By default, the noise is added following the Ornstein-Uhlenbeck process [44].

4. Applying DRL to the CW optimization

In order to apply DRL algorithms (DQN and DDPG) to the optimization of Wi-Fi networks, we propose a centralized approach to solving the CW optimization in this work. Our proposed approach consists of a centralized algorithm (i.e., the agent), which is a module running on the Wi-Fi access point (AP) that observes the state of the network (i.e., the environment) and chooses suitable CW values (i.e., the actions) to optimize the network's performance (i.e., the reward).

The agent is decided to run on the AP since it has a general view of the whole network and then can control the stations associated with it through beacon frames in a centralized way.

The current state, s , of the environment is the status of all stations associated with the AP. So, it is impossible to get this information because of the nature of the problem. Therefore, we model the problem as a partially observable Markov decision process (POMDP) instead of an MDP one. POMDP assumes the environment's state cannot be perfectly observed [45].

This proposal's adopted action, a , corresponds to 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. Therefore, the CW value to be broadcast to the stations can be obtained through the application of (2). This interval is selected so that the action space is within 802.11 standard's CW range, which ranges from 15 up to 1023. An action, a , taken by the agent in the state, s , makes the environment switch to its next state, s' , with a given transition probability, $T(s'|s, a)$.

$$CW = \left\lfloor 2^{a+4} \right\rfloor - 1 \quad (2)$$

The normalized network throughput, which can be observed at the AP, is used as the agent's reward, r , by taking action, a , in the state, s . Therefore, the reward is a real value in the interval $[0, 1]$. This normalized metric is obtained by dividing the actual throughput by the expected maximum.

For the observation, o , the normalized level of the transmission queue of each station, Q_{NL} , is used, according to (3). It is normalized by the maximum queue size value of each station. The normalized level of the transmission queue is adopted as the observation because it offers the best possible information about the network status.

$$Q_{NL} = \frac{Q_l}{Q_{max}} \quad (3)$$

The measurement of Q_{NL} is carried out at predefined intervals and indicates the result of the currently chosen CW value on the network's performance. For example, a value close to 1 indicates the queue is full, meaning the station cannot transmit packets as quickly as it receives them. On the other hand, if it is close to 0, the queue is almost empty, indicating the station can access the medium as frequently as necessary. A high Q_{NL} value indicates a high number of collisions. Conversely, a low value indicates a small number of collisions. The normalized level of the transmission queue of each station, Q_{NL} , is concatenated to data frames sent to the AP so that the agent has access to this information. At the AP, the agent normalizes the sum of Q_{NL} coming from the stations by the total number of stations associated with the AP.

4.1. Centralized DRL-based CW Optimization Method

The proposed method has three stages. The first one is a pre-learning stage, where the legacy Wi-Fi contention-based mechanism manages the network. This stage is used to initialize the DRL algorithm being used (either DQN or DDPG). Next, in the learning stage, the agent chooses CW values (i.e., actions) according to what is shown in Algorithm 1.

The mean, μ , and variance, σ^2 , of the history of recently observed normalized queue levels are calculated as a preprocessing step. Moving average with window and stride of fixed sizes is used to calculate both statistics. This calculation renders the observation into a two-dimensional vector for each stride of the moving average. Therefore, the agent is trained based on this two-dimensional vector of observations.

Exploration of the environment is enabled by adding a noisy factor to each action the agent takes. This noisy factor decreases throughout the learning stage. This addition of noise is different for each of the two considered DRL algorithms. When DQN is used, the noisy factor corresponds to the probability of taking a random action instead of an action predicted by the agent. In DDPG's case, the noisy factor comes from a Gaussian-distributed random variable and is added directly to the action taken by the agent. As mentioned, this is done to find a trade-off between exploring the environment and exploiting the acquired knowledge, which chooses the action that maximizes future rewards.

The last stage is called the operational stage. This stage starts when the training is over. The user defines the training stage's period. At this stage, the noisy factor is null, so the agent always chooses the action it learned to maximize the reward. At this stage, as the agent has already been trained, it does not receive any additional updates to its policy, so rewards are unnecessary.

Finally, it is essential to mention that the hyperparameters of the DRL models (i.e., learning rate, reward discount rate, batch size, epsilon decay) need to be fine-tuned for the agent to achieve its optimal performance. Lastly, as both DQN and DDPG employ reply memory, a size limit has to be configured for this memory buffer. The reply memory stores past interactions of the agent with the environment, i.e., it records the current state, the action taken at that state, the reward received in that state, and the next state resulting from the action taken. When its limit is reached, the oldest record is overwritten by a new one (i.e., it is implemented as a circular buffer).

Algorithm 1 DRL-based CW Optimization

```

1: Initialize the observation buffer,  $O$ , with zeroes
2: Initialize the weights,  $\theta$ , of the agent
3: Obtain the action function,  $A_\theta$ , of the agent
4: Define  $N_{RP}$  as the number of received packets
5: Get the queue level of each station
6: Specify  $envStepTime$  as the period of interaction with the environment
7: Define  $trainingFlag$  as a flag to tell the algorithm is in the training stage
8: Initialize the experience replay buffer,  $E$ 
9:
10:  $lastUpdate \leftarrow currentTime$ 
11:  $s \leftarrow$  vector of zeros
12:  $CW \leftarrow 15$ 
13:
14: for  $t = 1, \dots, \infty$  do
15:    $O.append(Q_{NL})$ 
16:   if  $lastUpdate + envStepTime \leq currentTime$  then
17:      $observation \leftarrow preprocess(O)$ 
18:      $a \leftarrow A_\theta(observation)$ 
19:      $CW \leftarrow 2^{a+4} - 1$ 
20:     if  $trainingFlag == \text{True}$  then
21:        $tput \leftarrow \frac{N_{RP}}{envStepTime}$ 
22:        $r \leftarrow normalize(tput)$ 
23:        $E.append((observation, a, r, s))$ 
24:        $s \leftarrow observation$ 
25:        $mb \leftarrow$  get mini-batch samples from  $E$ 
26:       Update  $\theta$  based on  $mb$ 
27:     end if
28:   end if
29: end for

```

Table 1. 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
Channel BW	Frequency 5 GHZ
Traffic	20 MHz
MCS	constant bit-rate UDP
Guard Interval	HeMcs (1024-QAM with a 5/6 coding rate)
Propagation delay model	800 [ns]
Propagation loss model	ConstantSpeedPropagationDelayModel
Simulation time	MatrixPropagationLossModel
	10 [s]

4.2. Experimentation Scenario

The proposed centralized DRL-based CW optimization solution is implemented on NS3-gym [16], which runs on top of the NS-3 simulator [15]. NS3-gym enables the communication between NS-3 (c++) and OpenAI gym framework (python) [46]. NS-3 is a network simulator based on discrete events mainly intended for academic research. It contains the implementation of several wired and wireless network standards [15]. In this work, we use version NS-3.29 of the NS-3 simulator. The DRL algorithms used here were implemented with tensorflow and pytorch.

The system model considered in this work is depicted in Figure 4. We consider a linear topology comprised of one AP and several stations transmitting packets. The AP plays the role of the DRL agent, selecting a new CW value according to the current observation. The stations send data packets to the AP, and the deployment of the stations can happen statically or dynamically. So, two scenarios are considered, one with static topology and the other with dynamic topology.

Table 1 presents the NS-3 parameters necessary to create the environment in which the agent will learn. Apart from those parameters, we assume single-user transmissions, a packet load adjusted to saturate the network, instant and faultless transference of network information, i.e., Q_{NL} , to the DRL agent, and that each station receives the selected CW value instantly. The last two assumptions allow the assessment of the proposed solution in an idealized scenario before going to more realistic ones. Realistic scenarios will require the transmission of Q_{NL} from each station to the AP and the periodic broadcast of the chosen CW to all stations through beacon frames. The only differences we foresee in the results presented here are a slower convergence of the agent and a slightly smaller throughput due to the transmission of the required overhead, i.e., Q_{NL} and CW.

Table 2 presents the agent parameters used in NS3-gym for the experiments. These parameters were empirically found through several simulations. The network architecture used by both DRL algorithms has one recurrent long short-term memory layer and two fully connected layers leading to an $8 \times 128 \times 64$ topology. The simulation experiment was executed for 15 episodes of 60 seconds. This configuration was also found by employing cross-validation strategies. The recurrent long short-term memory layer allows the DRL algorithms to consider past observations when predicting the best action in a given state. The moving average window is half the size of the observation history memory, and the stride is a fourth of its size.

Each one of the experiments consisted of 15 executions of 60-second long simulations, where the first 15 executions were part of the training stage, and the last one was the operational stage. Each one of the simulations consisted of 10 [ms] interaction intervals. Algorithm 1 was run between these interaction intervals.

Table 2. 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
Replay memory size	18,000
Size of observation history memory	300
envStepTime (i.e., interaction interval)	10 [ms]

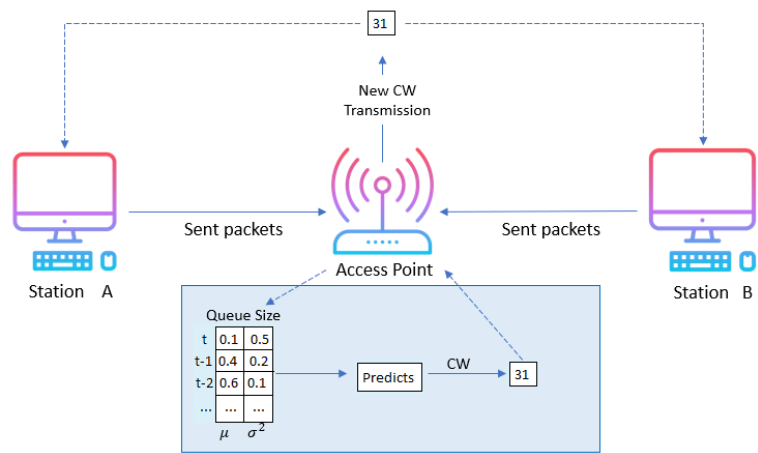


Figure 4. Scenario for assessing the proposed centralized DRL-based CW optimization method.

5. Simulation Results

This section presents and discusses the results obtained during the experiments. The performance of the proposed DRL algorithms is compared against the BEB algorithm, which is used in 802.11 wireless networks. Simulations were executed on NS-3 and NS3-gym simulators considering static and dynamic scenarios. The graphical results allow a better evaluation of the network efficiency achieved by the proposed centralized DRL-based CW optimization method in both scenarios. Next, we separate the results and discussion into two sets, static and dynamic.

5.1. Static scenario

In this scenario, the number of stations associated with the AP is kept constant throughout the experiment. As it is a static scenario, the optimal CW value should be constant, and so should the throughput. Therefore, this scenario is used to prove this hypothesis and to evaluate possible improvements over 802.11’s BEB algorithm.

Figure 5 shows the throughput achieved by the network for different numbers of stations. As can be seen, the network throughput decreases when BEB is employed. On the other hand, when either DQN or DDPG is used, it remains practically constant as the number of stations increases, proving the hypothesis. The improvement over BEB varies between 5.22% for 5 stations and up to 46.66% for 50 stations. DDPG has a slightly better performance than DQN, which can be explained due to its capability to choose any real CW value within the range [0,6].

Figure 6 shows the CW mean value for 15 simulation episodes when DQN or DDPG is used. This experiment considers 30 stations in the static scenario. It is pretty clear that the 14 episodes selected for the learning stage are adequate for the proposed solution to converge to an optimal and practically constant CW value for both DRL algorithms. It is also possible to see that the CW variance decreases along the learning stage, meaning that initially, the algorithm explores the environment more (i.e., it takes uncharted actions).

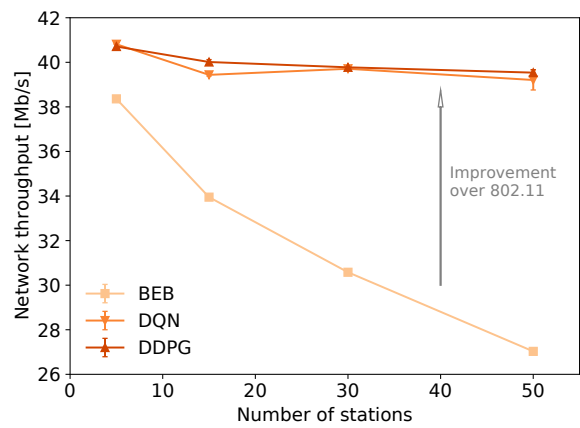


Figure 5. Comparison of the network throughput for the static scenario.

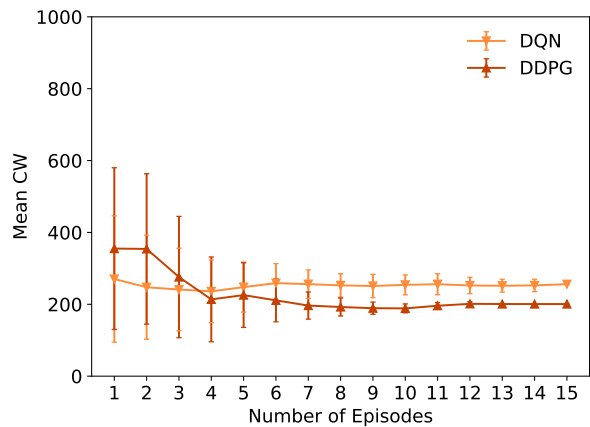


Figure 6. Mean CW value for 30 stations in the static scenario.

Then, as the number of episodes progresses, it exploits more of the acquired knowledge, which maximizes the received reward. Finally, the variance during the final episodes of the learning stage is small because the proposed algorithm correctly selected the CW value, which maximizes the throughput.

5.2. Dynamic scenario

In the dynamic scenario, the number of stations increases progressively throughout the experiment, going from 5 to 50. The higher the number of stations, the higher the collision probability. This experiment assessed whether the DRL algorithms appropriately act upon network changes.

Figure 7 depicts the chosen CW value for the dynamic scenario, where the number of stations progressively increases from 5 to 50. As shown, the chosen CW value increases as the number of stations increases, meaning the back-off interval has to be increased to accommodate the transmissions of the higher number of stations, mitigating the number of collisions. As can be noticed, DQN jumps between discrete neighbor CW values. At the same time, DDPG continuously increases the CW value, reaching a lower CW value for 50 stations, which positively reflects on the attained throughput.

Figure 8 compares the instantaneous network throughput in the dynamic scenario when the number of stations grows from 5 to 50. The increased number of stations alters the CW value, impacting the instantaneous network throughput. When the number of stations associated with the AP reaches 50, the throughput of the BEB drops to approximately 27% of that presented by DQN and DDPG. The proposed DRL algorithm (with either DQN or DDPG) presents an almost constant behavior, keeping a high and stable throughput as the

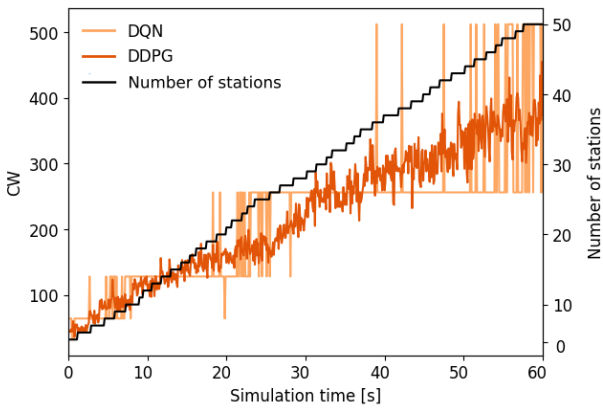


Figure 7. Selected CW value for different numbers of stations in dynamic scenario.

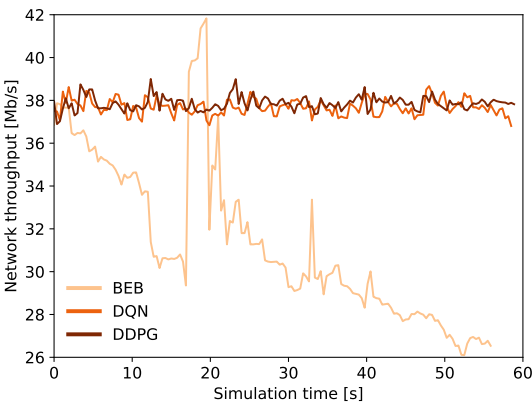


Figure 8. Comparison of the instantaneous network throughput as the number of stations increases from 5 to 50.

number of stations progressively increases. Both DRL algorithms present an approximately constant throughput of around 38.7 Mb/s.

Finally, figure 9 shows that both DRL algorithms improve the network’s throughput compared to the BEB algorithm. The improvement varies between 6.05% for 5 stations and up to 23.75% for 50 stations.

6. Conclusions

Wireless network transmissions are prone to various impairments (e.g., interference, path loss, channel noise, etc.) that lead to packet loss and collisions, making retransmission and channel access mechanisms required. Furthermore, in environments with a dense number of stations, more collisions will occur while the stations attempt to access the wireless channel. Consequently, the network efficiency and channel utilization will both degrade. This work proposes a centralized solution that employs DRL algorithms (i.e., DQN and DDPG) to optimize the CW parameter from the MAC layer. Regarding the number of stations associated with the AP, two experimental scenarios are considered for assessing the proposed centralized DRL-based CW optimization solution: static and dynamic. Simulation results show that the proposed solution outperforms the 802.11 default BEB algorithm by maintaining a stable throughput while reducing collisions. Moreover, the results attest to DQN’s and DDPG’s superior performance compared to BEB for both scenarios, regardless of the number of stations associated with the AP. The difference was amplified as the number of stations increased, with DQN and DDPG showing a 27% increase in throughput with 50 stations compared to BEB. Furthermore, DQN and DDPG had similar performances. Furthermore, the presented results show that the network’s

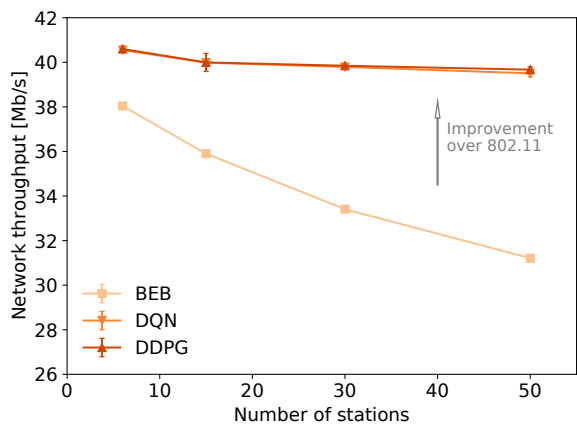


Figure 9. Comparison of the network throughput for the dynamic scenario.

performance can be dramatically improved when CW is chosen based on information from the network, such as the level of the transmission queues. It is also shown that a centralized solution for selection CW outperforms decentralized ones, such as the standard 802.11 BEB algorithm.

Future work could focus on using other ML algorithms to optimize CW, such as Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO). These two algorithms make an interesting topic of investigation because they both use advantage instead of Q-Value as the operator, with the difference between them being that SAC is off-policy and PPO is on-policy. A study that included these methods could be interesting because DQN and DDPG are both off-policy and use Q-value operator, which would contrast with SAC and PPO, potentially providing precious insights into the effect of these different operators in the final result.

Funding: This work was funded by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) via grant number 2070.01.0004709/2021-28 and partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) and by RNP, with resources from MCTIC, Grant No. 01250.075413/2018-04, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações - CRR) project of the National Institute of Telecommunications, Brazil; by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/EEA/50008/2020; and by the Brazilian National Council for Research and Development (CNPq) via Grant No. 313036/2020-9.

References

1. Guo, X.; Wang, S.; Zhou, H.; Xu, J.; Ling, Y.; Cui, J. Performance evaluation of the networks with Wi-Fi based TDMA coexisting with CSMA/CA. *Wireless Personal Communications* **2020**, *114*, 1763–1783.

2. Auzinger, W.; Obelovska, K.; Dronyuk, I.; Pelekh, K.; Stolyarchuk, R. A Continuous Model for States in CSMA/CA-Based Wireless Local Networks Derived from State Transition Diagrams. In Proceedings of the Proceedings of International Conference on Data Science and Applications. Springer, 2022, pp. 571–579.

3. Wang, G.; Qin, Y. MAC protocols for wireless mesh networks with multi-beam antennas: A survey. In Proceedings of the Future of Information and Communication Conference. Springer, 2019, pp. 117–142.

4. Beheshtifard, Z.; Meybodi, M.R. An adaptive channel assignment in wireless mesh network: the learning automata approach. *Computers & Electrical Engineering* **2018**, *72*, 79–91.

5. Wang, S.C.; Helmy, A. Performance limits and analysis of contention-based IEEE 802.11 MAC. In Proceedings of the Proceedings. 2006 31st IEEE Conference on Local Computer Networks. IEEE, 2006, pp. 418–425.

6. Yazid, M.; Sahki, N.; Bouallouche-Medjhoune, L.; Aïssani, D. Modeling and performance study of the packet fragmentation in an IEEE 802.11 e-EDCA network over fading channel. *Multimedia Tools and Applications* **2015**, *74*, 9507–9527.

7. Edalat, Y.; Obraczka, K. Dynamically Tuning IEEE 802.11's Contention Window Using Machine Learning. In Proceedings of the Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2019, pp. 19–26.

8. Bjornson, E.; Giselsson, P. Two applications of deep learning in the physical layer of communication systems [lecture notes]. *IEEE Signal Processing Magazine* **2020**, *37*, 134–140.

9. Anouar, H.; Bonnet, C. Optimal constant-window backoff scheme for IEEE 802.11 DCF in single-hop wireless networks under finite load conditions. *Wireless Personal Communications* **2007**, *43*, 1583–1602.

10. Bender, M.A.; Fineman, J.T.; Gilbert, S.; Young, M. How to scale exponential backoff: Constant throughput, polylog access attempts, and robustness. In Proceedings of the Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2016, pp. 636–654.

11. Al-Ammal, H.; Goldberg, L.A.; MacKenzie, P. Binary exponential backoff is stable for high arrival rates. In Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science. Springer, 2000, pp. 169–180.

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

13. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning, 2015. <https://doi.org/10.48550/ARXIV.1509.02971>.

14. Wydmański, W.; Szott, S. Contention window optimization in IEEE 802.11 ax networks with deep reinforcement learning. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2021, pp. 1–6.

15. Riley, G.F.; Henderson, T.R. The ns-3 network simulator. In *Modeling and tools for network simulation*; Springer, 2010; pp. 15–34.

16. Gawłowicz, P.; Zubow, A. Ns-3 meets OpenAI gym: The playground for machine learning in networking research. In Proceedings of the Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2019, pp. 113–120.

17. Abyaneh, A.H.Y.; Hirzallah, M.; Krunz, M. Intelligent-CW: AI-based framework for controlling contention window in WLANs. In Proceedings of the 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN). IEEE, 2019, pp. 1–10.

18. Xiao, Y.; Hirzallah, M.; Krunz, M. Distributed Resource Allocation for Network Slicing Over Licensed and Unlicensed Bands. *IEEE Journal on Selected Areas in Communications* **2018**, *36*, 2260–2274. <https://doi.org/10.1109/JSAC.2018.2869964>.

19. Kumar, A.; Verma, G.; Rao, C.; Swami, A.; Segarra, S. Adaptive contention window design using deep q-learning. In Proceedings of the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 4950–4954.

20. Fu, X.; Modiano, E. Learning-NUM: Network Utility Maximization with Unknown Utility Functions and Queueing Delay. In Proceedings of the Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, 2021, pp. 21–30.

21. Qureshi, I.A.; Asghar, S. A genetic fuzzy contention window optimization approach for IEEE 802.11 WLANs. *Wireless Networks* **2021**, *27*, 2323–2336.

22. Saini, T.K.; Sharma, S.C. Prominent unicast routing protocols for Mobile Ad hoc Networks: Criterion, classification, and key attributes. *Ad Hoc Networks* **2019**, *89*, 58–77.

23. Zerguine, N.; Mostefai, M.; Aliouat, Z.; Slimani, Y. Intelligent CW Selection Mechanism Based on Q-Learning (MISQ). *Ingénierie des Systèmes d'Inf.* **2020**, *25*, 803–811.

24. Ke, C.H.; Astuti, L. Applying Deep Reinforcement Learning to Improve Throughput and Reduce Collision Rate in IEEE 802.11 Networks. *KSII Transactions on Internet and Information Systems (TIIS)* **2022**, *16*, 334–349.

25. Mennes, R.; De Figueiredo, F.A.P.; Latré, S. Multi-Agent Deep Learning for Multi-Channel Access in Slotted Wireless Networks. *IEEE Access* **2020**, *8*, 95032–95045. <https://doi.org/10.1109/ACCESS.2020.2995456>.

26. Mennes, R.; Claeys, M.; De Figueiredo, F.A.P.; Jabandžić, I.; Moerman, I.; Latré, S. Deep Learning-Based Spectrum Prediction Collision Avoidance for Hybrid Wireless Environments. *IEEE Access* **2019**, *7*, 45818–45830. <https://doi.org/10.1109/ACCESS.2019.2909398>.

27. Ouameur, M.A.; Lê, D.T.A.; Massicotte, D.; Jeon, G.; de Figueiredo, F.A.P. Adversarial Bandit Approach for RIS-Aided OFDM Communication **2022**. 526
527

28. Aragão, M.V.C.; Mafra, S.B.; de Figueiredo, F.A.P. Otimizando o Treinamento e a Topologia de um Decodificador de Canal baseado em Redes Neurais. *Polar*, **2**, 1. 528
529

29. Adib Yaghmaie, F.; Ljung, L. A Crash Course on Reinforcement Learning. *arXiv e-prints* **2021**, pp. arXiv–2103. 530

30. Kibria, M.G.; Nguyen, K.; Villardi, G.P.; Zhao, O.; Ishizu, K.; Kojima, F. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE access* **2018**, *6*, 32328–32338. 531
532

31. Chen, M.; Challita, U.; Saad, W.; Yin, C.; Debbah, M. Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks. *arXiv preprint arXiv:1710.02913* **2017**, *9*. 533
534

32. Yang, H.; Alphones, A.; Xiong, Z.; Niyato, D.; Zhao, J.; Wu, K. Artificial-intelligence-enabled intelligent 6G networks. *IEEE Network* **2020**, *34*, 272–280. 535
536

33. Brewka, G. Artificial intelligence—a modern approach by Stuart Russell and Peter Norvig, Prentice Hall. Series in Artificial Intelligence, Englewood Cliffs, NJ. *The Knowledge Engineering Review* **1996**, *11*, 78–79. 537
538

34. Burnetas, A.N.; Katehakis, M.N. Optimal adaptive policies for Markov decision processes. *Mathematics of Operations Research* **1997**, *22*, 222–255. 539
540

35. Tokic, M.; Palm, G. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In Proceedings of the Annual conference on artificial intelligence. Springer, 2011, pp. 335–346. 541
542

36. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *nature* **2015**, *518*, 529–533. 543
544

37. Strehl, A.L.; Li, L.; Wiewiora, E.; Langford, J.; Littman, M.L. PAC model-free reinforcement learning. In Proceedings of the Proceedings of the 23rd international conference on Machine learning, 2006, pp. 881–888. 545
546

38. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* **2017**. 547
548

39. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2016, Vol. 30. 549
550

40. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* **2015**. 551

41. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* **2015**. 552
553

42. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International conference on machine learning. PMLR, 2014, pp. 387–395. 554
555

43. Konda, V.; Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems* **1999**, *12*. 556

44. Uhlenbeck, G.E.; Ornstein, L.S. On the theory of the Brownian motion. *Physical review* **1930**, *36*, 823. 557

45. Cassandra, A.R. A survey of POMDP applications. In Proceedings of the Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes, 1998, Vol. 1724. 558
559

46. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540* **2016**. 560
561