Enhanced Centralized Contention Window Optimization with Deep Reinforcement Learning on Highly Dense Wireless Networks

Hua-Ching Chen[1], Guang-Jhe Lin[2], Chih-Heng Ke[2]*

[1] School of Information Engineering, Xiamen Ocean Vocational College, China
[2] Department of Computer Science and Information Engineering, National Quemoy University, Taiwan
galaxy.km@gmail.com, guangjhe.lin@gmail.com, smallko@gmail.com

Abstract

With the continuous development and widespread adoption of wireless networks, there is an increasing demand for wireless network performance in highly dense network environments. The contention window (CW) value plays a crucial role in wireless network performance. However, the traditional Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism using the binary exponential backoff (BEB) algorithm cannot timely set appropriate CW values for highly dense networks. Therefore, we proposed an innovative CW control method called Enhanced Centralized Contention Window Optimization with DRL (ECCOD). This method dynamically controlled the CW values using deep reinforcement learning (DRL) to adapt to different network load levels. What sets ECCOD apart from other related studies is its ability to choose different CW adjustment modes to handle rapidly changing network conditions. This includes linearly increasing or maintaining the CW value, with further fine-tuning after decision-making. Simulation experiments demonstrate that different adjustment modes can achieve more fine-grained control, as expected. Specifically, our method outperforms CCOD (Centralized Contention Window Optimization with DRL) in terms of throughput performance under lightly loaded network conditions. In heavily loaded network conditions, our method achieves an average throughput improvement of 2.34% compared to CCOD, and surpasses the traditional CSMA/CA method by 38.59%.

Keywords: WLAN, CSMA/CA, Reinforcement learning, DQN, DRQN

1 Introduction

Wireless networks have become an indispensable part of our lives in today's society. With the continuous development and widespread adoption of wireless networks [1], the demand for wireless connectivity has increased considerably. However, there are still some challenges in the practical application of wireless networks, and one common issue is the occurrence of collisions during transmission. Collisions can reduce communication efficiency and impact the overall stability of the network.

Media Access Control (MAC) controls wireless signal transmission, including power regulation, rate adjustment, and determining transmission timing. Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) addresses collisions in wireless networks. The Contention Window (CW) determines transmission timing in CSMA/CA. When a node intends to transmit packets, it must wait for a specific duration when the communication channel is not in use. This waiting period, known as backoff (BO) time, is crucial for efficient packet transmission. The BO value selection depends on the CW. In the event of a collision or successful transmission, the CW value undergoes adjustments based on the backoff algorithm. CW can be adjusted for different environments to improve performance. High congestion requires increased CW range to reduce collisions, while low congestion allows for reduced CW range to enhance efficiency. However, excessively large CW values cause delays, while small values result in frequent collisions. Therefore, adjusting CW according to the network environment is crucial for optimal performance. In CSMA/CA, the CWmin and CWmax parameters are used to limit the range of CW, in order to adjust transmission time and performance in different environments.

The Binary Exponential Backoff (BEB) is utilized in the IEEE 802.11 standard to reduce collisions between nodes by exponentially increasing the CW value. As soon as a collision occurs, the CW value will exponentially increase to a larger value, and the node will choose its waiting time from that until the CW value reaches its maximum value. After a successful transmission, the node sets the CW value to the minimum. However, the BEB algorithm has limitations in adjusting the appropriate CW quickly due to the correlation between network conditions in consecutive time units. It requires significant time to readjust CW after each successful transmission. Furthermore, in densely populated node environments, the exponential growth of CW in the BEB algorithm can lead to performance issues [2]. As the number of nodes increases, the BEB algorithm continuously adjusts CW exponentially to handle high network loads and prevent collisions. However, CW exponential growth can become excessive, particularly in densely populated and heavily loaded network environments. This results in excessively long waiting times for nodes, wasting idle resources and

*Corresponding Author: Chih-Heng Ke; Email: smallko@gmail.com
DOI: https://doi.org/10.70003/160792642026072704006

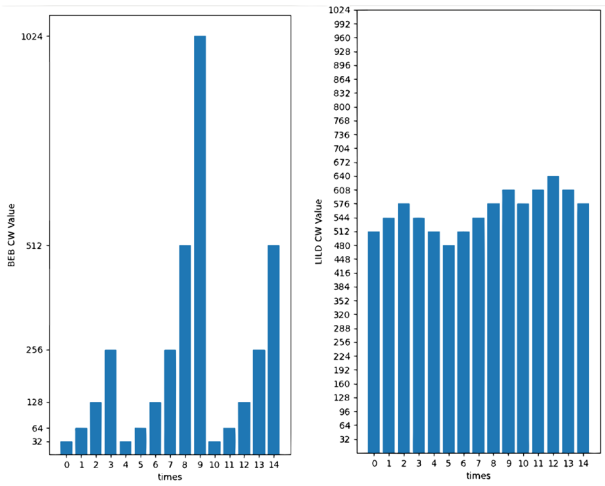
ultimately affecting network throughput [3].

Common method for controlling the CW. Linear Increase, Linear Decrease (LILD) has advantages over BEB when dealing with large numbers of network nodes, as it adjusts the CW linearly instead of exponentially. This approach is more effective at adapting to the network environment, especially in scenarios with high node counts and relatively small environmental changes. In recent years, with advancements in artificial intelligence technologies [4], several studies [5-13] have utilized machine learning to optimize CW. Some studies, such as [8-13], employ a centralized management approach where an Access Point (AP) acts as an intelligent agent for reinforcement learning (RL). This avoids the limitations of running RL computations on different Mobile Station (MS). However, compared to distributed architectures [2-4, 13-15], this approach requires frequent broadcasting of decisions by the AP, which consumes network resources and may decrease system throughput. In highly dense wireless networks, the performance of CW adjustment mechanisms will be further challenged. However, the existing CCOD method is still unable to finely control the CW. CCOD can only select a relatively appropriate value from a limited set of CW values to control transmission, and this limitation may lead to congestion in highly dense scenarios.

Therefore, this study proposes the use of deep reinforcement learning (DRL) to quickly find the appropriate CW value for the current scenario. CW values can also be fine-tuned according to transmission success or failure in order to increase transmission efficiency. This approach enhances wireless network performance. Experimental results demonstrate significant improvements in throughput and a reduction in collisions. This study provides novel insights and methods for optimizing wireless network performance. The main contributions of this research are as follows: proposing an effective method based on the Adaptive Increment-Decrement Algorithm for enhancing wireless network performance, improving existing algorithms to adapt to complex network topologies while increasing computational speed, validating the proposed method's effectiveness and practicality through experiments, and offering new ideas and methods for performance optimization in wireless networks. The paper is structured as follows: Section 2 reviews related work, Section 3 presents the problem statement, Section 4 describes the Adaptive Increment-Decrement Algorithm, Section 5 explains the experimental scenarios and methodology, Section 6 showcases the experimental results and provides discussions, and finally, the conclusion is presented.

2 Background and Related Works

This study focuses on Distributed Access Control issue in wireless sensor networks. Among them, BEB and LILD mechanisms are two commonly used approaches for ensuring fairness when multiple nodes compete for resources. BEB typically employs exponential increase and reset operations for the CW. LILD performs linear increase and decrease operations with a certain step size, aiming for smoother resource allocation. The operation modes of both mechanisms are illustrated in Figure 1, where it can be observed that LILD searches for an appropriate CW value more smoothly.



(a) BEB (b) LILD

Figure 1. Operation modes of BEB and LILD

In addition, in recent years, the frequency of applying machine learning techniques in wireless network-related research has been increasing [7]. Machine learning methods have also received attention in addressing the Distributed Access Control problem. Among them, neural network techniques combined with deep learning have been widely used for predicting and optimizing resource allocation. For example, Y. Edalat et al. [5] designed a simple and effective algorithm based on the Fixed-Share approach to adjust the contention window of IEEE 802.11 for significant performance improvement. To the extent of the researchers' knowledge, they were the first to propose a machine learning algorithm that automatically adjusts the IEEE 802.11 contention window based on the history of data packet transmissions.

J. Xin et al. [6] proposed an intelligent wireless MAC protocol based on deep learning. By designing a deep neural network (DNN), the protocol takes historical Received Signal Strength Indication (RSSI) as input and outputs joint channel access and rate adaptation decisions. It addresses the performance issues of existing Wi-Fi network media access control protocols (CSMA/CA) in high-density deployments.

In addition to machine learning methods, RL has also emerged as a popular solution. This allows systems to learn autonomously and adjust and optimize their strategies. In particular, RL is through continuous interaction with the environment to explore optimal action combinations in various scenarios. RL characterizes the current environment using a state space, which is the network state in this study. The set of all possible actions that can be attempted in different network states is referred to as the action space. The RL agent gains insights into the benefits of specific operations in a given state through the rewards

obtained by selecting actions. RL computes reward values through a reward function. Thus, the design of the reward function plays a crucial role in determining the learning direction of the agent.

Furthermore, the feedback from the agent regarding the action incorporates not only the immediate reward values but also weighs considerations for the future benefits of the action. As depicted in Equation (1), it is used to update the agent's strategy. Where γ represents the weight assigned to future returns and $max_a Q(s', a')$ is the estimate optimal Q-value of next state s'. α signifies the emphasis on feedback for the current action selection, in other words, it is learning rate. Where s denotes the state, a denotes the action, $R(s, a)$ denotes the reward function, and $Q(s, a)$ denotes the value after a comprehensive assessment of the action taken. The agent can maintain a table, known as the Q-table Q, in tabular form to record the Q-values $Q(s, a)$ associated with executing different actions in various states.

$$Q'(s, a) = Q(s, a) + \alpha \cdot [R(s, a) + \gamma \cdot max_a Q(s', a') - Q(s, a)] \quad (1)$$

However, when confronted with intricate network environments, the storage space required to save the Q-table becomes excessively large. To tackle this issue, the Deep Q-Network (DQN), a DRL technique, was developed to approximate the Q-table using neural networks. The agent inputs the state into neural networks, and neural networks output the Q value of each action. The function is similar to the Q table. This innovation aims to circumvent the challenge of managing extensive storage requirements in complex network environments.

Wydmański et al. [8] proposed a centralized contention window optimization scheme using DRL (CCOD). In CCOD, the AP acts as a DRL agent and selects suitable contention window values based on collision rates, periodically sending revised CW values through beacon frames.

Sanada et al. [9] presented a simple RL-based method for adjusting the IEEE 802.11 contention window. Their approach uses a single action, selecting the minimum CW value, as the action space. Simulated experimental results showed that the proposed method is simpler than traditional approaches but provides nearly the same network performance.

Lee et al. [10] employed the AP as a RL agent to learn and adjust CW, broadcasting the CW values to all users to avoid hardware limitations. Extensive evaluations demonstrated that their proposed method significantly improved Wi-Fi uplink performance in terms of throughput and channel access delay, and it could dynamically adjust its parameters in response to changes in the network state.

However, all of these methods focus on optimizing the contention window, and system performance can only be guaranteed when decision-making is re-implemented. This heavily relies on decision implementation frequency, which may limit their effectiveness in large-scale network environments. Overall, this study explores various mechanisms and methods for addressing the Distributed Access Control problem. It proposes a more effective and fair resource allocation scheme.

3 Problem Statement

With the rapid growth of Wi-Fi users and mobile traffic, especially in high-density usage scenarios such as airports, conference centers, and schools, capacity and performance issues in Wi-Fi networks have become increasingly severe. Traditional Wi-Fi networks employ the contention-based Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) transmission scheme, which can lead to network congestion and inefficiency. Therefore, improving Wi-Fi networks' capacity and performance has become a critical topic in current Wi-Fi technology research. In this study, we will explore an emerging Wi-Fi technology that effectively addresses existing bottlenecks and issues in current technologies, enhancing network capacity and performance to better meet different scenarios.

In the CSMA/CA strategy, the CW size setting directly impacts Wi-Fi networks' throughput and latency. If the CW is too small, numerous collisions occur under high loads, resulting in low network efficiency, decreased throughput, and increased latency. On the other hand, if the CW is too large, waiting time and network response time increase, affecting the user experience. However, the BEB algorithm used in the IEEE 802.11 standard causes the CW to exponentially increase and then suddenly fall to its minimum value.

This CW control is not very effective, leading to drastic variations in CW values and the need for more frequent CW adjustments. This leads to decreased system throughput. Additionally, the LILD method avoids drastic changes by linearly adjusting the CW value, but the smaller adjustment range also limits its adaptability to dynamic network changes. Therefore, a method capable of quickly adjusting the CW value to adapt to network dynamics is still needed. However, the emergence of the CCOD algorithm, which combines DRL techniques, effectively addresses the limitations of the BEB algorithm. This algorithm can intelligently and rapidly select the appropriate CW under current network loads, reducing transmission collisions and idle time, and improving system throughput.

We propose an effective method that further expands the system's action space based on CCOD. By combining different CW adjustment modes, we can maintain a certain operational space for MS's CW even during each interval when the intelligent agent reselects actions. This achieves more refined control. Our proposed solution will provide better network performance results.

4 Proposed Scheme

4.1 System Model

In traditional distributed network architectures, when the BEB or LILD algorithms are used for the first time

in an unfamiliar environment, their CW values start from the minimum CW value and are modified through trial and error to adapt to the current network load. However, in a traditional distributed network architecture, all nodes continuously adjust their CW values to adapt to the same network conditions. The adjustment of CW values will impact the network state with delay. Therefore, in a decentralized CW management network, whenever network conditions change, there is a constant need to adjust the CW values. This situation becomes more challenging in network environments with a large number of nodes, as these algorithms require more retries to find the appropriate CW values due to their inherent limitations. To address this problem, we employ a centralized CW control architecture and introduce DRL to improve network performance.

In this study, we propose a system architecture as shown in Figure 2. This architecture places the intelligent agent within the wireless AP. When each node transmits data to the AP, we use a piggybacking method to enable the AP to understand the current network environment. The intelligent agent utilizes the ECCOD algorithm (see Algorithm 1) to collect data from each node for training to find the most suitable CW value for the given environment. Once the intelligent agent completes training, it can determine the appropriate CW value based on the current node's transmission data. It can then broadcast the updated CW value to all nodes through beacons for updating.

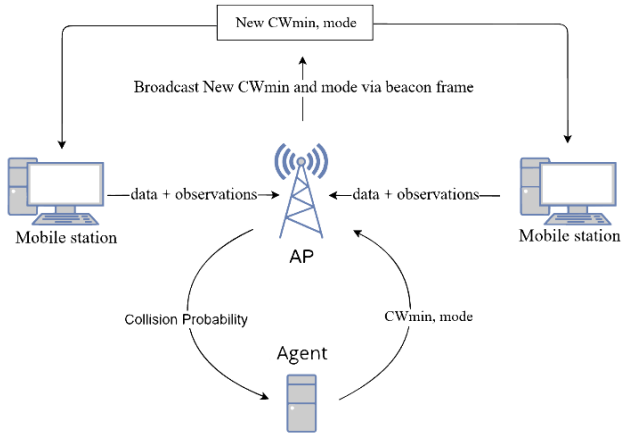


Figure 2. System architecture diagram

4.2 Action Space

At the initial stage, the agent collects data from each node through the AP for training. Based on the CW adjustment modes we set, is as Equation (2), actions can be categorized into two types: the first type is to keep the CW value unchanged regardless of successful or failed packet transmissions, indicating that the current CW value is optimal for the current environment and does not require adjustment. The second type is to linearly increase the CW value when packet transmission fails, aiming to reduce collision rates. Since the initial environment is unknown to the intelligent agent, we use a random approach to determine actions in different states and obtain rewards.

$$A_t = \begin{cases} Linearly\ increase\ CW, & when\ packet\ failure \\ Keep\ unchanged, & suitable\ or\ successful \end{cases} \quad (2)$$

Here, A_t represents the action at time t, and CW represents the Contention Window. This represents the waiting time interval for the next transmission in transmission collisions. When packet transmission fails, CW increases, indicating a decrease in available system bandwidth. When the current CW is suitable for the current environment, it remains unchanged. In an unknown environment, the agent randomly determines actions to obtain rewards.

In traditional RL approaches, the action space is typically defined as a range of options that grow from the minimum to the maximum values of a wireless network protocol, such as {32, 64, 128, 256, 512, 1024}. Within the RL framework, these options are mapped sequentially to actions ranging from 0 to 5. However, we introduced a mode selection mechanism that increases the original action space dimensionality from 6 to 12. In other words, the RL agent makes decisions by selecting an action from the range of 0 to 11. Each option represents a different CW value in a specific mode.

Specifically, when the action value is between 0 and 5, it indicates mode 0. The CW options are {32, 64, 128, 256, 512, 1024}. On the other hand, when the action value is between 6 and 11, it represents mode 1. The CW options remain [32, 64, 128, 256, 512, 1024}. Here, mode=0 indicates that regardless of success or failure, the CWmin and CWmax of the node are reset to the CW value calculated by the algorithm. In contrast, mode=1 signifies the need to fine-tune the CW value when transmission fails. Initially, the algorithm-determined CW value is set to the node's CWmin. If the transmission fails, the CW value is increased by a fixed value, *linear_increment*, which is set to 16 in this study. If the transfer is successful, the CW is reset to the original CW value determined by the algorithm.

4.3 Observation Space

In this experiment, we use collision rate as the numerical evaluation of the observed environment, denoted as N_{coll}. The number of successfully received packets is represented as S_s, and the number of collided packets is represented as S_c. The collision rate can be calculated by dividing the total number of failed transmitted packets by the sum of successful and failed transmitted packets. This is shown in the following equation.

$$N_{coll} = \frac{S_c}{S_s + S_c} \quad (3)$$

4.4 Reward Function

Based on Equation (3), we can calculate the collision rate for each state. Through the collision rate, we can evaluate whether the chosen action in each state has a positive impact on the current network performance. Next,

we will explain the reward design used in this experiment. After training, we aim to find actions that achieve the highest throughput in different states. Therefore, we set the throughput as the reward value and let the agent choose the current optimal action based on the influence of the reward value. However, since our desired outcome is the optimal solution after future training, the reward function incorporates a discount factor γ ($0 < \gamma < 1$) to reduce the impact of each occurrence of local optimal solutions on the agent. A smaller value of γ, closer to 0, implies that the agent places more emphasis on immediate rewards and prioritizes short-term gains. On the other hand, a larger value of γ, closer to 1, signifies that the agent values long-term rewards and takes into account the cumulative impact of future actions. With the discount factor adjusted, we can optimize for long-term performance while considering immediate rewards.

4.5 ECCOD Algorithm

To optimize CW values and improve network transmission efficiency, this paper proposes the ECCOD algorithm. ECCOD learns the optimal initial CW value and selects appropriate adjustment modes to enhance throughput and reduce collisions.

In environments with many nodes, the AP's agent interacts with the environment, observes the current state, and determines the best action. All nodes are then set to the same CW size, ensuring fairness and improving transmission efficiency. The specific implementation of the ECCOD algorithm, including the selected optimal CWmin and adjustment modes, is described in Algorithm 1. Table 1 also describes relevant parameter descriptions.

Algorithm 1 Pseudocode for ECCOD-DQN Algorithm

1: **procedure** LEARNING($H_{(N_{coll})}$, $load$, a)
2: $s \leftarrow$ preprocess($H_{(N_{coll})}$)
3: $r \leftarrow$ normalize($load$)
4: agent.step(s, a, r) ▷ Train a neural network
5: $a \leftarrow$ agent.act(s)
6: CW_{min},mode$\leftarrow$ SetCW(a)
7: **return** CW_{min}, $mode$
8: **end procedure**
9: **procedure** EVALUATION($H_{(N_{coll})}$)
10: $s \leftarrow$ preprocess($H_{(N_{coll})}$)
11: $a \leftarrow$ agent.act(s) ▷ Send to the neural network
12: CW_{min},mode$\leftarrow$ SetCW(a)
13: **return** CW_{min}, $mode$
14: **end procedure**
15: **procedure** CONTROL_CW(CW_{min}, mode)
16: **for** $node$ **in** $node\_set$ **do**
17: **if** successful transmission **then**
18: $node.cw \leftarrow CW_{\text{min}}$
19: **else if** transmission failure **then**
20: **if** mode $= 0$ **then**
21: $node.cw \leftarrow CW_{\text{min}}$
22: **end if**
23: **if** mode $= 1$ **then**
24: $node.cw \leftarrow node.cw + linear\_increment$
25: **end if**
26: **end if**
27: **end for**
28: **end procedure**

The procedure LEARNING(.) is responsible for training the neural network model. Once the model is trained, the procedure EVALUATION(.) accesses the model to generate CW adjustment decisions for the nodes. On the node side, LEARNING(.) adjusts the CW values based on the decisions provided, ensuring that the configuration is dynamically optimized according to network conditions. In lines 20 to 24 of the algorithm, we adaptively choose whether to allow a linear increase in the CW value based on network conditions, providing flexibility.

Table 1. Parameters description

Parameter	Description
CW_{min}	Minimum contention window
load	Data sent since the last interaction
$H_{(N_{coll})}$	Recent Observed Collision Rate History
a	Action
s	State
agent.act	Choose an action
SetCW	Choose the way to modify according to the action
$node\_set$	Set of all the nodes

5 Results and Discussion

5.1 Settings

We implemented the ECCOD algorithm proposed in this paper using the DQN algorithm. To compare the advantages of the ECCOD algorithm and the CCOD algorithm with the traditional CSMA/CA algorithm, we implemented the ECCOD algorithm using the PFRL framework. We used CCOD with DQN for validation. Additionally, Deep Recurrent Q-Network [16] (DRQN), a variant of DQN, was employed in the ECCOD algorithm. DRQN is potentially more suitable than DQN for handling partially observable environments with temporal dependencies. In wireless networks, the transmission and reception of signals are often influenced by past states and actions. Thus, employing DRQN allows for better capturing of these temporal dependencies, enhancing the performance and stability of the algorithm. We validated the ECCOD algorithm model in the research methodology by simulating feasibility and performance in a wireless network using Python. The simulator parameters served as references for network parameters. We further analyzed and simulated the ECCOD algorithm's performance in a wireless network. The analysis model and simulation parameters used in this experiment are detailed in Table 2. We chose the PFRL framework because it supports various RL algorithms and runs efficiently on both CPU and GPU. This enables the rapid and effective implementation of the ECCOD algorithm and simulation analysis. For the simulation experiment network parameters, we adopted the IEEE 802.11b parameter settings. The specific parameter values used in our study are detailed in Table 2.

Table 2. Network simulation parameters

Parameter	Value
Packet payload	1000 Bytes
Channel bit rate	11 Mbps
Slot time	20 μs
DIFS	50 μs
SIFS	10 μs
CWmin	32
CWmax	1024
linear_increment	16

Table 3 presents the ECCOD algorithm parameters. We set the memory size to 1,000,000 to store states, actions, rewards, and next-stage states during training. The agent utilizes stored data for training, and during each training iteration, the agent selects 32 samples for training. The proposed algorithms in this paper employ a neural network with three layers. The first and second layers consist of 128 neurons each, and ReLU is used as the activation function. The size of the third layer corresponds to the action space. Each value represents a specific action in the action space.

Table 3. DRL parameters

Parameter	Value
Memory size	1,000,000
Memory warmup size	200
Batch size	32
Learning rate	0.001
Discount factor	0.9
Epsilon greedy	0.1

5.2 Convergence Performance

To ensure the ECCOD method's convergence, we evaluated the impact of different training durations on the model's loss within a network of 90 nodes. For ECCOD with DQN, Figure 3, approximately 50,000 seconds of simulation time were required for the model to converge. In contrast, Figure 4, ECCOD with DRQN achieved convergence in only about 200 seconds. This significant difference in convergence time is due to DRQN's ability to effectively handle temporal dependencies and sequential data.

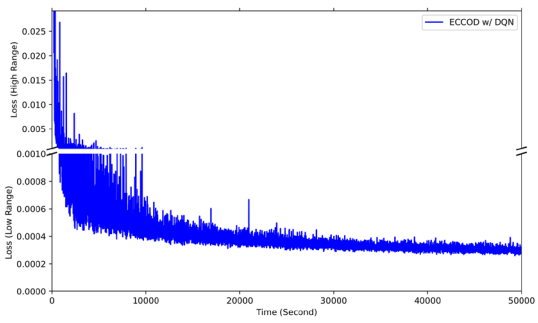


Figure 3. Convergence performance of ECCOD with DQN over 50,000 seconds

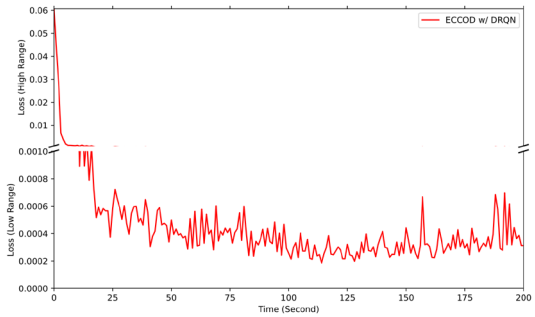


Figure 4. Convergence performance of ECCOD with DRQN over 200 seconds

By comparing Figure 3 and Figure 4, it is evident that while both DQN and DRQN achieve convergence, DRQN demonstrates a faster initial convergence compared to DQN. The rapid stabilization of loss in DRQN highlights its efficiency in handling temporal dependencies and sequential data more effectively than DQN. This comparative analysis underscores the potential advantages of utilizing DRQN in environments where the sequence of observations is critical for decision-making.

5.3 Static Topology

The experiments were conducted across networks comprising nodes ranging from 30 to 90, representing various network scales, including small, medium, and large-scale scenarios. During these experiments, we compared the performance of the traditional legacy-CSMA/CA algorithm with the CCOD-DQN algorithm and our proposed approach. The main focus was on throughput and collision rate. Figure 5 and Figure 6 present the experimental results.

In terms of collision rates, as depicted in Figure 5, the ECCOD method outperforms both the CCOD algorithm and traditional CSMA/CA, achieving significantly better results. Specifically, in networks with 30 to 90 nodes, ECCOD with DQN achieved collision rates of 9.17%, 10.45%, 11.07%, and 11.70%, respectively. Meanwhile, ECCOD with DRQN achieved even lower collision rates of 5.02%, 9.21%, 9.27%, and 11.61%, respectively. As expected, ECCOD with DRQN showed superior performance.

On average, ECCOD with DRQN exhibited a collision rate reduction of 13.08% compared to CCOD and 69.02% compared to CSMA/CA. This significant improvement is attributed to ECCOD's more precise control over the CW values for each node, resulting in enhanced performance compared to CCOD.

Figure 6 illustrates that the normalized throughput results mirror the trends observed in collision rates. Throughput is a direct measure of the efficiency of different algorithms because it accounts for the actual amount of data successfully transmitted by all nodes, factoring in the impact of collisions. For networks with 30 to 90 nodes, ECCOD with DRQN achieved standardized throughput values higher than CCOD by 2.24%, 2.99%, 3.08%, and 2.34%. On average, these values surpass those of CCOD

and CSMA/CA by 2.66% and 29.36%, respectively.

The experimental results demonstrate that the ECCOD algorithm effectively reduces the collision rate across various node counts. It also performs well in terms of normalized throughput. These findings highlight the significant advantages of the ECCOD algorithm in enhancing network communication efficiency.

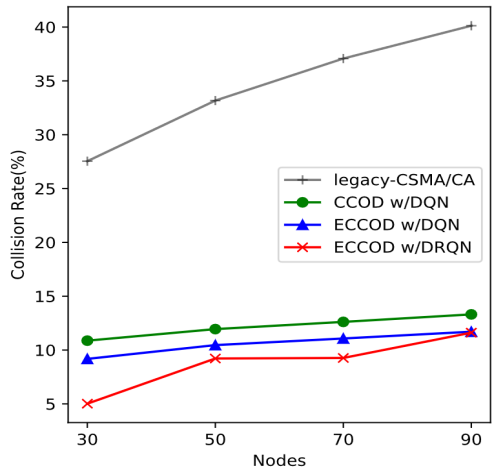


Figure 5. Comparing experimental results with collision rate

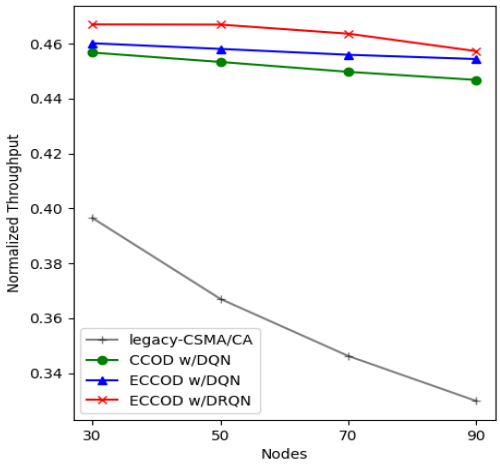


Figure 6. Comparing experimental results with normalized throughput

6 Conclusion

This study proposes ECCOD, a method combining the Adaptive Increment-Decrement algorithm and CCOD, to address wireless network collisions and enhance performance. We implemented ECCOD using DQN and DRQN techniques, with experimental results showing that DRQN converges faster and outperforms DQN. The effectiveness of ECCOD is validated through experiments, presenting novel optimization approaches. ECCOD's learning capability allows it to be applied to different IEEE 802.11 APs by adjusting parameters. Simulations demonstrate that with 90 nodes, ECCOD-DRQN outperforms CCOD by 2.34% and achieves 38.59% higher throughput compared to CSMA/CA, extending CCOD's method to various CW adjustment modes.

In future work, deploying decentralized multi-agent designs with RL agents on each node could reduce the computational load of centralized processing while offering differentiated services. Additionally, using more realistic environments, such as NS-3, could simulate real-world conditions, such as coexistence between nodes with different transfer rates, to further test our method in practical scenarios.

References

[1] R. Wood, S. P. Bonacci, *Wireless Network Data Traffic: Worldwide Trends and Forecasts 2021–2026*, Analysys Mason, October, 2021.

[2] L. Zhang, H. Yin, Z. Zhou, S. Roy, Y. Sun, *Enhancing WiFi Multiple Access Performance with Federated Deep Reinforcement Learning*, arXiv:2102.07019, February, 2021. https://doi.org/10.48550/arXiv.2102.07019

[3] B.-J. Kwak, N.-O. Song, L. E. Miller, Performance analysis of exponential backoff, *IEEE/ACM Transactions on Networking*, Vol. 13, No. 2, pp. 343-355, April, 2005. https://doi.org/10.1109/TNET.2005.845533

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, *A Brief Survey of Deep Reinforcement Learning*, arXiv:1708.05866, August, 2017. https://doi.org/10.48550/arXiv.1708.05866

[5] Y. Edalat, K. Obraczka, Dynamically Tuning IEEE 802.11's Contention Window Using Machine Learning, *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '19)*, Miami Beach, FL, USA, 2019, pp. 19-26. https://doi.org/10.1145/3345768.3355920

[6] J. Xin, W. Xu, Y. Cai, T. Wang, S. Zhang, P. Liu, Z. Guo, J. Luo, Deep Learning Based MAC via Joint Channel Access and Rate Adaptation, *2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, Helsinki, Finland, 2022, pp. 1-7. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860617

[7] S. Szott, K. Kosek-Szott, P. Gawowicz, J. T. Gomez, B. Bellalta, A. Zubow, F. Dressler, Wi-fi meets ml: A survey on improving IEEE 802.11 performance with machine learning, *IEEE Communications Surveys & Tutorials*, Vol. 24, No. 3, pp. 1843–1893, Third quarter, 2022. https://doi.org/10.1109/COMST.2022.3179242

[8] W. Wydmaski, S. Szott, Contention Window Optimization in IEEE 802.11ax Networks with Deep Reinforcement Learning, *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, 2021, pp. 1-6. https://doi.org/10.1109/WCNC49053.2021.9417575

[9] K. Sanada, H. Hatano, K. Mori, Simple Reinforcement Learning based Contention Windows Adjustment for IEEE 802.11 Networks, *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2023, pp. 692-693. https://doi.org/10.1109/CCNC51644.2023.10059960

[10] K.-H. Lee, D. Kim, CFX: Contention-Free Channel Access for IEEE 802.11ax, *Sensors*, Vol. 22, No. 23, Article No.

9114, December, 2022.
https://doi.org/10.3390/s22239114

[11] Y. Yu, T. Wang, S. C. Liew, Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks, *2018 IEEE International Conference on Communications (ICC)*, Kansas, MO, USA, 2018, pp. 1-7.
https://doi.org/10.1109/ICC.2018.8422168

[12] A. Pressas, Z. Sheng, F. Ali, D. Tian, A q-learning approach with collective contention estimation for bandwidth-efficient and fair access control in IEEE 802.11p vehicular networks, *IEEE Transactions on Vehicular Technology*, Vol. 68, No. 9, pp. 9136–9150, September, 2019.
https://doi.org/10.1109/TVT.2019.2929035

[13] C. Wu, S. Ohzahata, Y. Ji, T. Kato, A MAC protocol for delay-sensitive VANET applications with self-learning contention scheme, *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, Las Vegas, USA, 2014, pp. 438–443.
https://doi.org/10.1109/CCNC.2014.6866607

[14] K. Kosek-Szott, A. Lo Valvo, S. Szott, P. Gallo, I. Tinnirello, Downlink channel access performance of NR-U: Impact of numerology and mini-slots on coexistence with Wi-Fi in the 5 GHz band, *Computer Networks*, Vol. 195, Article No. 108188, August, 2021.
https://doi.org/10.1016/j.comnet.2021.108188

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*, arXiv:1602.05629, v4, January, 2023.
https://doi.org/10.48550/arXiv.1602.05629

[16] M. Shivani, S. Rao, C. N. Sujatha, A Survey on Deep Recurrent Q Networks, in: S. Nandan Mohanty, V. Garcia Diaz, G. A. E. Satish Kumar (Eds.), *Intelligent Systems and Machine Learning. ICISML 2022*, Springer, 2023.
https://doi.org/10.1007/978-3-031-35078-8_21

Biographies

Hua-Ching Chen received the Ph.D. degrees in Electronic Engineering from Xiamen University, Xiamen, Fujian, China, P.R.C., in 2014. He is currently the lecturer with the Department of School of Information Engineering, Xiamen Ocean Vocational College. His current research interests include wireless networks, optimal learning algorithms and image processing.

Guang-Jhe Lin is currently working toward the B.S. degree in computer science and information engineering at the National Quemoy University, Kinmen, Taiwan. His research interests focus on software defined networks and reinforcement learning.

Chih-Heng Ke received his B.S. and Ph.D. degrees in electrical engineering from National Cheng-Kung University, in 1999 and 2007. He is a professor at the Department of Computer Science and Information Engineering in National Quemoy University, Kinmen, Taiwan. His research interests include wireless networks, software defined networks, and reinforcement learning.