Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

Enhancing IEEE 802.11 Standard with Deep Reinforcement Learning for Optimal Channel Access

SHEILA CÁSSIA DA SILVA JANOTA CRUZ

JUNHO / 2024



ENHANCING IEEE 802.11 STANDARD WITH DEEP REINFORCEMENT LEARNING FOR OPTIMAL CHANNEL ACCESS

SHEILA CÁSSIA DA SILVA JANOTA CRUZ

Thesis presented to the National Institute of Telecommunications as part of the requirements for obtaining the Master's degree in Telecommunications.

SUPERVISOR: Felipe Augusto Pereira de Figueiredo.

Santa Rita do Sapucaí
2024

Cruz, Sheila Cássia da Silva Janota
C957e.

Enhancing IEEE 802.11 Standard with Deep Reinforcement Learning for Optimal Channel Access./ Sheila Cássia da Silva Janota Cruz – Santa Rita do Sapucaí, 2024.

91 p.

Orientador: Prof. Dr. Felipe Augusto Pereira de Figueiredo
Dissertação de Mestrado em Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.

Inclui bibliografia.

1. Wi-Fi 2.Contention-based access scheme 3. Channel utilization optimization 4. Machine learning 5. Reinforcement learning 6. Mestrado em Telecomunicações. I. Figueiredo, Felipe Augusto Pereira de . II. Instituto Nacional de Telecomunicações – INATEL. IV. Título.

CDU 621.39

Ficha Catalográfica elaborada pelo Bibliotecário da Instituição
CRB6-2718 – Paulo Eduardo de Almeida

Biblioteca "Ministro Olavo Bilac Pinto"

FOLHA DE APROVAÇÃO

Thesis presented and approved on _____/_____/_____, by the judging committee:

Prof. Dr. Felipe Augusto Pereira de Figueiredo
INATEL

Prof. Dr. Samuel Baraldi Mafra
INATEL

Prof. Dr. Joel José Puga Coelho Rodrigues
UFPI

Coordenador do Curso de Mestrado
Prof. Dr. José Marcos Camara Brito

“Ouça os conselhos e esteja pronto para aprender; assim um dia você será sábio”

Provérbios 19.20

Aos meus pais, meu irmão, meu esposo e filho,
podem ter acerteza que são tudo para mim.

Acknowledgments

First and foremost, I thank you, Gracious Heavenly Father, for the breath of life and for always being by my side leading my path.

I express my heartfelt gratitude to the Ministry of Telecommunications, Information Technologies of Angola, and Social Communication for the scholarship concession. And a special thanks to José Carvalho da Rocha, the former Minister of Telecommunications, Information Technologies of Angola, and Social Communication, as well as Américo dos Santos, the Director of the ”Fundo de apoio às Comunicações.”

I am deeply grateful to my supervisor, Felipe Augusto Pereira de Figueiredo, for his mentorship using methodologies that could make me learn and for his expertise and patience throughout this entire process. His valuable insights and constant encouragement were indispensable in shaping the direction of this dissertation.

I would like to give thanks to Gisele Moreira dos Santos, from the Inatel academic records service. For the wonderful woman that she is, for being very professional in what she does, and even more. When I was going through a rough moment, we talked, and just like a mother, she gave me a warm hug that I needed. Thanks, and may God bless you always.

I am sincerely thankful to the National Institute of Telecommunications - Inatel, Brazil, for providing resources such as well-equipped laboratories, library facilities, and an academic environment that enriched my academic experience. To all the Professors for the valuable lessons that motivated me to acquire more knowledge. A special thanks to Professors José Brito, Samuel Mafra and Edson Gimenez.

I thank my friends and colleagues Solival Zito, Vitor Figueiredo, Elvira Diogo, Aline Aires, Marcelo Cysneiros, Eduardo Teixeira, Danilo de Oliveira, Alcides Mário, Elivander Pereira, Mateus Cruz, Vignon Fidèle, Gustavo Moreira and Diego Zúñiga for the great talk and experience exchange.

I give thanks to all my relatives, including brothers, sisters, uncles, nephews, and my mother-in-law, for their advice and moral support. A special thanks goes to my beloved parents, Gonçalo Janota and Catarina Janota. They were my first teachers and taught me that nothing is impossible if we put in hard work and have faith in God. Thanks, Mom and Dad, for all your prayers. May God bless both of you.

I extend my appreciation to my son’s babysitter, Rafaella, for taking such great care of him while I study at Inatel. May God bless you always.

I express my heartfelt gratitude to my husband, Mauro Cruz, for his love and dedication and for always pushing me to be better and become a better version of myself. I thank God for giving me such an amazing man and awesome dad that you are. Without your support, I couldn’t get this far. I love you very much.

A special thanks to my son Marco Rafael. Even though he is just one year old, he has taught me so much. Thank you, son, for giving me the strength to never give up and to keep moving forward.

Contents

Contents x

List of Figures xiii

List of Tables xv

Abbreviations and Acronyms xviii

Resumo xxi

Abstract xxiii

1 Introduction 1
- 1.1 Motivation . . . 6
- 1.2 Problem Statement . . . 7
- 1.3 Research Objectives . . . 8
- 1.4 Contributions . . . 8
- 1.5 Document Organization . . . 9
- 1.6 Publications . . . 10

2 Theoretical Background and Literature Review 11
- 2.1 Machine Learning Overview . . . 11
  - 2.1.1 Supervised Learning . . . 11
  - 2.1.2 Unsupervised Learning . . . 12
  - 2.1.3 Reinforcement Learning . . . 13
  - 2.1.4 Deep Reinforcement Learning . . . 18
  - 2.1.5 Multi-Agent Reinforcement Learning . . . 25
- 2.2 Literature Review . . . 28

3 Proposed Methodology 35
  - 3.0.1 Problem formulation . . . 37
  - 3.0.2 Network topology . . . 45
  - 3.0.3 Observation Metrics . . . 45
  - 3.0.4 Reward Function . . . 47

4 Results and Analysis 49
  - 4.0.1 Simulation setup . . . 49

4.0.2 Centralized Solution Results 51
4.0.3 Decentralized Solution Validation 62
4.0.4 Centralized vs Decentralized Results 64

5 Conclusions and Future Work **75**
5.0.1 Chapter Summary 76
5.0.2 Final Remarks 78
5.0.3 Limitations and Future Work 79

6 Funding **81**

References **83**

List of Figures

1.1 Basic IEEE 802.11 DCF operation. 2
1.2 Binary Exponential Back-off. 3
1.3 Supervised vs unsupervised vs Reinforcement Learning 5

2.1 ML learning paradigms based on the type of training. 12
2.2 Markov Chain State Transition 14
2.3 Markov Decision Process State Transition 15
2.4 Markov Decision Process State Transition Matrix 15
2.5 An RL agent interacting with the environment. 16
2.6 Q-learning. 17
2.7 DRL Structure. 19
2.8 Deep Q-learning (DQN). 21
2.9 Deep Deterministic Policy Gradient (DDPG). 23
2.10 Multi-agent reinforcement learning environment. 26

3.1 Bi-dimensional observation vector used in the preprocessing phase. . 38
3.2 Scenario for assessing the proposed centralized DRL-based CW optimization method. 43
3.3 Scenario for assessing the decentralized DRL-based CW optimization method. 45

4.1 Architecture of the Deep Learning Network. 52
4.2 Comparison of the network throughput for the static centralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol and Avg queue level as observations. . . . 53
4.3 Mean CW value for 5 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 53
4.4 Mean CW value for 15 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 54
4.5 Mean CW value for 30 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 54
4.6 Mean CW value for 50 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 55
4.7 Mean CW value for 5 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 57
4.8 Mean CW value for 15 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 57

4.9 Mean CW value for 30 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 58
4.10 Mean CW value for 50 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. . . . 58
4.11 Selected CW value with the number of stations increasing from 5 to 50 in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations. 59
4.12 Comparison of the instantaneous network throughput in a dynamic centralized scenario using BEB, DQN, and DDPG with Pcol and Avg queue level as observations. 60
4.13 Comparison of the network throughput for the dynamic centralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol and Avg queue level as observations. . . . 61
4.14 Validation test comparing the instantaneous network throughput for the decentralized static and dynamic scenarios using different action types, including DQN, DDPG, random, and constant. 62
4.15 Validation test comparing the cumulative reward for the decentralized static and dynamic scenarios using different action types, including DQN, DDPG, random, and constant. 63
4.16 Validation test comparing the cumulative reward for the decentralized static and dynamic scenarios using DQN and DDPG. 64
4.17 Comparison of the network throughput for the static centralized and decentralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol as the observation. 65
4.18 Mean CW value for 5 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 66
4.19 Mean CW value for 15 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 66
4.20 Mean CW value for 30 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 67
4.21 Mean CW value for 50 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 67
4.22 Mean CW value for 5 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 68
4.23 Mean CW value for 15 in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 69
4.24 Mean CW value for 30 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation. . . 69
4.25 Mean CW value for 50 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation. . . 70
4.26 Mean CW value with the number of stations increasing from 5 to 50 in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation. 71
4.27 Comparison of the instantaneous network throughput in a dynamic centralized and decentralized scenario using BEB, DQN, and DDPG with Pcol as observation. 71

4.28 Comparison of the network throughput in a dynamic centralized and decentralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol as observation. 72

List of Tables

2.1 A summary of the related works using a centralized DRL approach. . 29
2.2 A summary of the related works using a decentralized DRL approach. 32

4.1 NS-3 Environment configuration parameters. 50
4.2 NS3-gym agent configuration parameters. 51

Abbreviations and Acronyms

ACK	Acknowledgment response
ALOHA	Additive links on-line Hawaii area
ANN	Artificial neural networks
AP	Access point
BS	Base station
BEB	Binary exponential backoff
CBR	Constant bit rate
CSMA/CA	Carrier-sensing multiple access with collision avoidance
CSI	Channel state information
CSIM	Discrete-event-based simulator
CW	Contention window
CWMax	Maximum contention window
CWMin	Minimum contention window
DCF	Distributed coordination function
DDPG	Deep deterministic policy gradient
DDQN	Double deep q-learning network
DIFS	Distributed interFrame space
DNN	Deep neural network
DQN	Deep q-network or Deep q-learning
DRL	Deep reinforcement learning
EWMA	Exponentially weighted moving average
GF-CWO	Genetic fuzzy-contention window optimization
GPU	Graphics processing units
HetNet	Heterogeneous network
HBAB	History-based adaptive backoff
IEEE	Institute of electrical and electronics engineers
KNN	K-nearest neighbors
LSTM	Long short-term memory
ML	Machine learning
MAC	Medium access control
MADDPG	Multi-agent deep deterministic policy gradient
MANETs	Mobile ad hoc networks
MARL	Multi-Agent reinforcement learning
MASC	Multiple-actor-shared-critic
MDP	Markov decision process
MCS	Modulation and coding scheme
NS-3	Discrete event network simulator

NS3-gym	Enables comm. between NS-3 and OpenAI Gym framework
O-DCB	On-demand channel bonding
OFDM	Orthogonal frequency-division multiplexing
PHY	Physical layer
PCF	Point coordination function
PPO	Proximal policy optimization
POMDP	Partially observable Markov decision process
RF	Radio frequency
SAC	Soft actor-critic
SETL-DQN	Smart exponential-threshold-linear backoff algorithm with DQN
SIFS	Short inter-frame space
SVM	Support vector machine
TPUs	Tensor processing units
TD3	Twin delayed deep deterministic policy gradient
UDP	User datagram protocol
Wi-Fi	Wireless fidelity
WLAN	Wireless local area network

Resumo

Com o avanço do campo tecnológico, os sistemas de comunicação sem fio aumentaram significativamente sua mobilidade e flexibilidade, permitindo melhor qualidade de serviço para um número cada vez maior de dispositivos conectados. As redes sem fio se tornaram parte integrante de nossas vidas diárias, suportando uma variedade de aplicações. No entanto, o aumento contínuo de dispositivos sem fio intensificou a necessidade de estratégias inteligentes para acessar com eficiência o espectro limitado disponível, especialmente em cenários densos que são propensos a colisões ou perda de pacotes quando os recursos do espectro são compartilhados de forma não justa. O mecanismo de prevenção de colisões adotado pelo padrão IEEE 802.11 não é ótimo, porque utiliza um algoritmo de retrocesso exponencial binário (BEB) na camada de controle de acesso ao meio (MAC). Tal algoritmo aumenta o intervalo de retrocesso sempre que uma colisão é detectada para minimizar a probabilidade de colisões subsequentes. No entanto, o aumento do intervalo de retrocesso causa degradação da utilização do espectro de rádio (ou seja, desperdício de largura de banda). Esse problema piora quando a rede precisa gerenciar o acesso ao canal para um grande número de estações, levando a uma diminuição dramática no desempenho da rede. Além disso, uma configuração incorreta de retrocesso aumenta a probabilidade de colisões, de modo que as estações experimentam numerosas colisões antes de atingir o valor ideal de retrocesso. Portanto, para mitigar o desperdício de largura de banda e, consequentemente, maximizar o desempenho da rede, este trabalho propõe o uso de algoritmos de aprendizado por reforço (RL), nomeadamente Deep Q-Network (DQN) e Deep Deterministic Policy Gradient (DDPG), para enfrentar esse problema de otimização. Na abordagem proposta, para além das métricas de observação (média do nível normalizado da fila de transmissão e probabilidade de colisões), também são avaliadas duas abordagens: a centralizada e a descentralizada. O throughput geral da rede é definido como a recompensa. A ação é o valor da janela de contenção (CW) que maximiza o throughput, minimizando o número de colisões. Quanto às simulações, o simulador de rede NS-3 é utilizado juntamente com uma ferramenta conhecida como NS3-gym, que integra um framework de RL no NS-3. Os resultados demonstram que DQN e DDPG têm desempenho muito melhor do que BEB tanto para cenários estáticos quanto dinâmicos, independentemente do número de estações. Os resultados mostram que observações baseadas na média do nível normalizado das filas de transmissão têm um desempenho ligeiramente melhor do que observações baseadas na probabilidade de colisões. Além disso, a diferença de desempenho com BEB é ampliada à medida que o número de estações aumenta, com DQN e DDPG apresentando um aumento de 45,52% no throughput com 50 estações na abordagem centralizada e um aumento de

37.16% na abordagem descentralizada. Vale ressaltar que DQN e DDPG apresentaram desempenhos semelhantes, o que significa que qualquer um deles poderia ser utilizado e quando aumenta o número de estações, a abordagem centralizada apresenta melhor performance em relação a descentralizada.

Palavras-Chave: Wi-Fi, esquema de acesso baseado em contenção, otimização da utilização do canal, aprendizado de máquina, aprendizado por reforço, NS-3, NS3-gym.

Abstract

With the advancement of the technological field, wireless communication systems have significantly enhanced their mobility and flexibility, enabling better quality of service for an ever-growing number of connected devices. Wireless networks have become an integral part of our daily lives, supporting a variety of applications. However, the ongoing increase in wireless devices has intensified the need for intelligent strategies to efficiently access the limited available spectrum, especially in dense scenarios that are prone to collisions or packet loss when the resources of the spectrum are unfairly shared. The collision avoidance mechanism adopted by the IEEE 802.11 standard is not optimal, because it 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 increase of the backoff interval causes degradation of the radio spectrum utilization (i.e., bandwidth wastage). The 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. Additionally, 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-Network (DQN) and Deep Deterministic Policy Gradient (DDPG), to address this optimization problem. In the proposed approach, besides observation metrics (average normalized transmission queue level and collision probability), two approaches are also evaluated: centralized and decentralized. The overall throughput of the network is defined as the reward. The action is the contention window (CW) value that maximizes throughput while minimizing the number of collisions. As for the simulations, the NS-3 network simulator is used along with a toolkit known as NS3-gym, which integrates an 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. Additionally, our results show that observations based on the average of the normalized level of the transmission queues have a slightly better performance than observations based on the collision probability. Moreover, the performance difference with BEB is amplified as the number of stations increases, with DQN and DDPG showing a 45.52% increase in throughput with 50 stations in the centralized approach and an increase of 37.16% in the decentralized approach. It is worth noting that DQN and DDPG exhibited similar performances, meaning that either of them could be used. As the number of stations increases, the centralized approach outperforms the decentralized one.

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

Chapter 1

Introduction

REcent technological advances have ushered in significant progress within the wireless local area networks (WLANs). Some notable enhancements in WLANs lie in providing better network performance, mobility, flexibility, quality of service, and efficient processing of vast amounts of data. These characteristics make the WLANs the most extensively used network [1] [2]. Furthermore, as the number of connected devices increases exponentially, there is a growing demand for WLAN devices to accommodate a diverse range of applications [3] [4], including voice, video streaming, access to the cloud, and traffic offloading meeting requirements such as quality of service.

The IEEE 802.11, or simply wireless fidelity (Wi-Fi), is a set of wireless network standards designed and maintained by the Institute of Electrical and Electronics Engineers (IEEE) [5] [6] that defines medium access control (MAC) and physical layer (PHY) protocols for deploying WLANs. Their MAC layer provides two different operation modes for accessing the wireless channel, namely point coordination function (PCF) and distributed coordination function (DCF) [7]. PCF is a centralized contention-free MAC protocol; here, the channel access is controlled by an access point (AP) by sending polling messages to the stations. DCF is a distributed contention-based MAC protocol, which is the most widely used in the industry for random access in scenarios with multiple stations. To understand how the DCF channel access occurs, Figure 1.1 presents the basic IEEE 802.11 DCF operation.

Initially, a station starts by sensing the medium status (i.e., idle or busy) before transmitting a data packet. If the medium status is idle during a Distributed Inter-Frame Space (DIFS) period, then the station proceeds with its transmission [8]. On the other hand, if the medium status is sensed as busy after a station has deferred for a DIFS period, then a station selects a random delay, also known as a back-off period, to allow

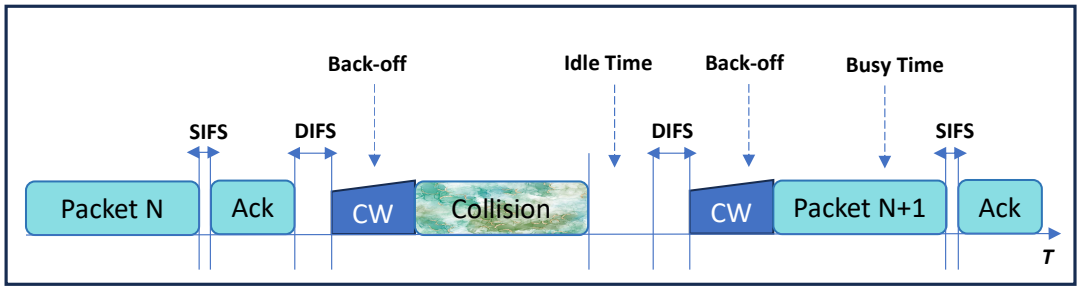

Figure 1.1: *Basic IEEE 802.11 DCF operation.*

the transmission [8]. After a station transmits its data packet, the transmitting station needs confirmation if the packet has been successfully received on the destination side, so the Short Inter-Frame Space (SIFS) period is utilized as a boundary between the data packet and the acknowledgment (ACK) response if the transmitting station does not receive this ACK response it may be presumed as a packet loss or collision and to solve this the random back-off mechanism is applied for retransmitting the data packet [9].

The IEEE 802.11 DCF 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 [10, 11]. With CSMA/CA, the nodes compete to access the radio resources and, consequently, the channel [12, 13]. 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 has its value doubled every time a collision occurs (e.g., when an ACK is 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 [14]. This deferring mechanism is also known as binary exponential back-off (BEB) [15] [16] and is shown in Figure 1.2.

The BEB algorithm has several limitations and may not always be the best solution for collision avoidance, often providing suboptimal solutions [17, 18]. These limitations include inefficiency under high loads, lack of fairness among competing nodes, inability to adapt to changing network conditions, vulnerability to the hidden node problem, and no global optimization. While the BEB algorithm is widely used in Wi-Fi networks, it may not consistently offer the best approach for avoiding collisions.

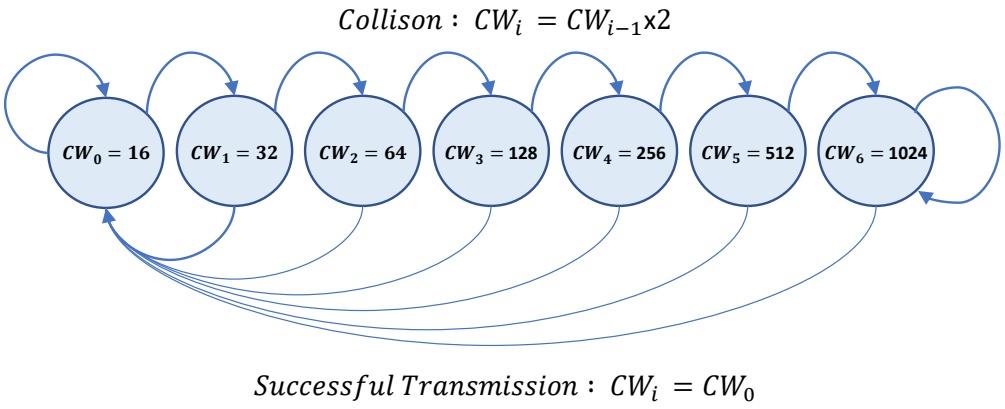

Figure 1.2: *Binary Exponential Back-off.*

Other approaches, such as machine learning-based ones, may be better able to address the limitations of the BEB algorithm and provide more effective collision avoidance strategies.

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 has its value doubled when collisions are detected, leading to an inefficient network operation [19].

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 [20]. Since network devices with high computational capabilities are becoming increasingly common, CW can be optimized through machine learning (ML) algorithms [3]. In ML, the most common learning paradigms are supervised, unsupervised, and reinforcement learning [21]. As illustrated in Figure 1.3, summarizes the difference between supervised versus unsupervised versus reinforcement learning.

Supervised algorithms require a labeled dataset where the outcomes for the respective inputs are known [22]. This learning paradigm is well-suited for tasks such as classification and regression, as it estimates values based on established patterns. In the context of Wi-Fi collision avoidance, supervised algorithms can be trained on datasets labeled with network traffic patterns, collision events, throughput rates, and the number of nodes in the system. These data points enable the algorithms to learn to identify conditions that are likely to cause collisions and proactively suggest actions to minimize them.

Unsupervised learning occurs when the ML algorithm is trained using unlabeled data [23]. The idea behind this paradigm is to find hidden (i.e., latent) patterns in the

data. This paradigm could be used to identify hidden patterns in Wi-Fi traffic that could be contributing to collisions. By clustering similar traffic patterns, it may be possible to identify common sources of interference or other issues that are leading to collisions and take action to mitigate them.

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 [24]. This paradigm allows the agents (e.g., Wi-Fi nodes) to explore the environment by taking random actions and finding optimal solutions. This paradigm can be employed to optimize the actions of Wi-Fi nodes in a network to minimize collisions and maximize throughput. The nodes can learn to take actions that lead to higher reward values (e.g., successful transmissions) while avoiding actions that lead to lower reward values (e.g., collisions). This paradigm presents several advantages for collision avoidance, including the ability to learn from experience, optimize long-term performance, handle complex and dynamic environments, explore and exploit different strategies, and adapt to different contexts. RL algorithms can learn from the state of the network and adapt to changing network conditions, optimizing cumulative rewards over time. This makes RL well-suited to handle the complexity of Wi-Fi networks, find optimal solutions, and navigate different scenarios. Additionally, RL is flexible and adaptable, making it a powerful tool for collision avoidance in Wi-Fi networks. Therefore, since there is no optimal solution for the CW optimization (BEB is known to be suboptimal) [25–27], RL may offer more effective collision avoidance strategies.

Despite their benefits, the learning paradigms present some limitations, for instance, when creating a dataset for supervised algorithms 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 [28]. Nonetheless, unsupervised learning can be limited in its applicability to collision avoidance in Wi-Fi networks due to the lack of labeled data, limited interpretability of results, limited control over the learning process, and lack of a feedback loop for ongoing adjustment and optimization. While unsupervised learning can be useful for analyzing Wi-Fi traffic data and identifying patterns, other machine learning paradigms, such as reinforcement learning, may be better suited to the problem of collision avoidance.

Despite the increasing state-of-the-art contributions presented, RL-based algorithms are complex, presenting high computational requirements, require fine-tuned hyperpa-

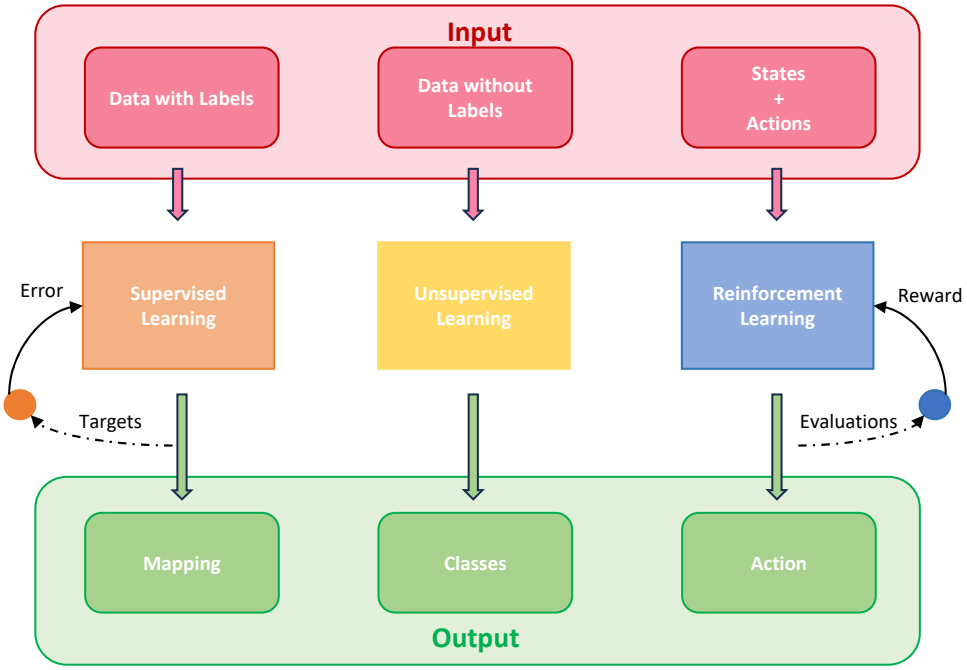

Figure 1.3: *Supervised vs unsupervised vs Reinforcement Learning*

rameters since they are sensitive to their settings, the training process might take several hours and need extensive exploration, requiring a significant amount of computing power, and present difficulties in handling continuous and high-dimensional state and action spaces [29]

RL algorithms have addressed their limitations through various approaches. To mitigate high computational requirements, techniques like parallelization (i.e., distributed computing) and hardware acceleration have been employed to leverage the power of multiple processors or machines, and hardware acceleration using specialized processors like graphics processing units (GPUs) or tensor processing units (TPUs) to expedite the training process [30]. To tackle the need for extensive training data and exploration, methods such as experience replay, where past experiences are stored and randomly sampled during training, allow for efficient and effective utilization of data [31]. Exploration strategies, such as epsilon-greedy or Thompson sampling, strike a balance between exploiting learned knowledge and exploring new possibilities [32].

Moreover, curiosity-driven or intrinsic motivation learning techniques have been explored to encourage agents to explore novel states or seek out new experiences, thereby reducing the reliance on vast amounts of explicit training data [33]. Sensitivity to hyperparameters is addressed through techniques like grid search, systematically exploring different hyperparameter combinations to find optimal settings [34]. More advanced approaches, such as Bayesian optimization, leverage probabilistic models to

search the hyperparameter space intelligently [35].

Furthermore, algorithmic advancements like automatic hyperparameter tuning methods, utilizing meta-learning or reinforcement learning itself, have been developed to automate the selection of hyperparameters, enhancing the performance of the algorithms [36]. Difficulties in handling continuous and high-dimensional spaces are overcome using deep neural networks and parameterization methods. Deep neural networks are used to function approximation, allowing for learning policies directly from raw sensory inputs [37]. Continuous action spaces are handled through parameterization methods, such as using Gaussian distributions or policy parameterization [38]. Additionally, actor-critic architectures, combining policy and value function approximators, have shown promise in effectively handling high-dimensional state and action spaces [39]. These ongoing advancements aim to improve the efficiency and performance of reinforcement learning algorithms across different domains.

1.1 Motivation

Wireless networks are widely used and applied in different domains where the stations connected to the wireless networks require a fair share of the spectrum resources to guarantee a better performance while accessing the channel to transmit data. A fair resource allocation means that every station connected to the network has an equitable share of the network resources. In practice, this means that each user or station has a reasonable opportunity to access or utilize network resources without the risk of a single user or group of users monopolizing the resources, resulting in a better experience for the users.

One of the challenges in achieving a fair resource allocation is addressing collisions, where multiple stations attempt to transmit data simultaneously, resulting in signal interference and potential data loss. Currently, one of the most popular ways to avoid collisions is through the BEB algorithm, but it has several limitations and often provides suboptimal solutions. The limitations and drawbacks present in the BEB algorithm highlight the need for an optimized solution to enhance the collision avoidance mechanism in IEEE 802.11 networks.

To address the issues of the BEB algorithm, this work proposes using deep reinforcement learning (DRL) algorithms to optimize the CW value selection and improve the performance of Wi-Fi networks by maintaining a stable throughput and minimizing collisions. More specifically, we propose an RL-aided centralized and decentralized solution aimed at finding the best CW value that maximizes the overall Wi-Fi through-

put of the network by properly setting CW values, especially in dense environments with dozens to hundreds of stations.

The proposed solution takes actions, i.e., selects a CW value based on the observation metric adopted. In this work, we study and compare the use of two distinct DRL algorithms, namely, DQN and DDPG, and two different observation metrics, namely, averaged normalized transmission queues level of all associated stations and the collision probability, to tackle the CW optimization problem. DQN was chosen because it is relatively simple and has a discrete action space. However, despite its simplicity, DQN generally displays performance and flexibility that rivals other methods [40]. DDPG was selected since it is a more complex method that represents actions as continuous values, yielding an exciting comparison with DQN [41]. By using DDPG, we want to explore the hypothesis that the proposed solution can adjust the CW value more precisely and in a more fine-grained fashion with a continuous action space.

1.2 Problem Statement

The expansion of IEEE 802.11 WLANs due to their flexibility is leading to an increasing number of wireless-connected devices, such as access points and mobile stations. These devices require guaranteed network performance and throughput even when operating in a congested scenario. Furthermore, spectrum utilization becomes a massive issue in dense scenarios because of the limited resources available that, most of the time, are not fairly shared between the wireless devices and wireless communication technologies that compete to access the radio channel, leading to the degradation of spectrum utilization affecting negatively the network performance, generating packet loss and collision.

The most used collision avoidance algorithm in IEEE 802.11 networks is the BEB algorithm. In this algorithm, when different stations are concurrently trying to access the medium, the CW (i.e., back-off period) value is exponentially doubled whenever a collision occurs, avoiding selecting the same CW value for all the stations. The process of duplicating the CW value delays the station transmission leading to various challenges and performance issues, such as bandwidth wastage, decreased network performance in dense networks, and scalability problems as the number of stations in the network increases. The incorrect back-off values generated by BEB increase collision occurrences drastically, making channel access very challenging.

1.3 Research Objectives

This thesis aims to propose RL-based algorithms to enhance the performance of IEEE 802.11 networks by the optimization of a chosen network parameter. Considering a centralized as well as a decentralized topology. Employing RL algorithms to improve network performance, reduce collision occurrences, and enhance scalability in dense and dynamic scenarios. By applying RL algorithms to the collision avoidance problem, we aim to find an optimized solution that outperforms the BEB algorithm. To reach the proposed goal, the following partial objectives were defined:

- Perform a comprehensive review regarding IEEE 802.11 networks and reinforcement learning algorithms.
- Identify which WLAN parameter should be optimized to improve the network efficiency and utility.
- Identify what information should be used to describe the network status better.
- Identify how the chosen parameters affect the wireless network performance.
- Identify Which RL algorithms can be used to optimize the chosen parameters.
- Compare the performance of the chosen algorithms to traditional collision avoidance algorithms in wireless networks.

1.4 Contributions

During the Thesis, Machine Learning was used to mitigate the number of collisions in Wi-Fi networks, both in dynamic and static scenarios involving up to 50 stations competing for network resources. In this sense, the main contributions of the thesis were as follows:

- An observation employing the value of the averaged normalized transmission queue level was utilized to analyze system performance, revealing that it provides a more accurate depiction of the network status. This study is detailed in Chapters 3 and 4 and has been published in the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME).
- A comparison between two different types of observation to train the agent, namely the averaged normalized transmission queue level and the probability of collision. This study is detailed in Chapter 4 and was published in the Journal of Communication and Information Systems.

- A decentralized simulator that is different than the approaches available in the literature was created using NS3-Gym. Where the observation of the agent uses the probability of collision, is presented in Chapters 3 and 4.

1.5 Document Organization

The remainder of this dissertation is organized as follows. Chapter 2 provides the theoretical background, offering a comprehensive overview of machine learning concepts, detailing its learning paradigms, the challenges they face, and the key algorithms used for optimizing network parameters in wireless networks. This chapter also includes a literature review, discussing various related works on centralized and decentralized solutions for collision avoidance in wireless networks. Chapter 3 outlines the proposed methodology, describing the input and output values related to the observation of the state, action, and reward of the agent, along with a detailed explanation of the proposed algorithms and the network topology used in the implementation. Chapter 4 presents the results and analysis of the centralized and decentralized approaches, including details of the simulation setup specifying the parameters used on the NS-3 simulator and NS3-gym framework. The results of the centralized approach focus on two different observation metrics, while the decentralized approach contrasts the centralized and decentralized solutions using the same observation metric. Chapter 5 concludes the dissertation by summarizing the key findings, and lessons learned, and providing future work recommendations.

1.6 Publications

Throughout this research, two scientific papers were published, one in an international Journal and another in an international conference. A third was accepted for publication in an national conference.

Publication in international Journal:

- **Sheila Cássia da Silva Janota Cruz**, Messaoud Ahmed Ouameur, and Felipe Augusto Pereira de Figueiredo, "Reinforcement Learning-based Wi-Fi Contention Window Optimization," Journal of Communication and Information Systems, vol. 38, no. 1, pp. 128–143, 2023. DOI: 10.14209/jcis.2023.15.

Publication in international Conference:

- **Sheila Cássia Da Silva Janota Cruz**, Felipe Augusto Pereira de Figueiredo and Messaoud Ahmed Ouameur, "Enhancing IEEE 802.11 Standard with Deep Reinforcement Learning for Optimal Channel Access, "2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)", Tenerife, Canary Islands, Spain, 2023, pp. 1-6, DOI: 10.1109/ICECCME57830.2023.10252258.

Accepted for publication to national Conferences:

- **Sheila Cássia Da Silva Janota Cruz**, Felipe Augusto Pereira de Figueiredo and Rausley Adriano Amaral de Souza, "Decentralized Deep Reinforcement Learning Approach for Channel Access Optimization," 2024 XLII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT), Belém, Pará, Brazil.

Chapter 2

Theoretical Background and Literature Review

2.1 Machine Learning Overview

ML algorithms, also called models, have been widely applied to solve different problems related to optimization in wireless network communications systems [42–45]. 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 [46,47]. 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 [48].

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 [49]. It is possible to classify the ML model learning based on the type of its training, also called learning paradigm [50]. The learning paradigms can be classified as **Supervised learning**, **Unsupervised learning**, and **Reinforcement Learning (RL)**. Figure 2.1 shows the relations between the ML paradigms. Therefore, next, we provide a brief overview of these learning paradigms.

2.1.1 Supervised Learning

In the supervised learning paradigm, the ML model uses labeled data during the training phase, where each input sample is accompanied by its desired output sample, called a label [51]. It is suitable for applications with plenty of historical data [52].

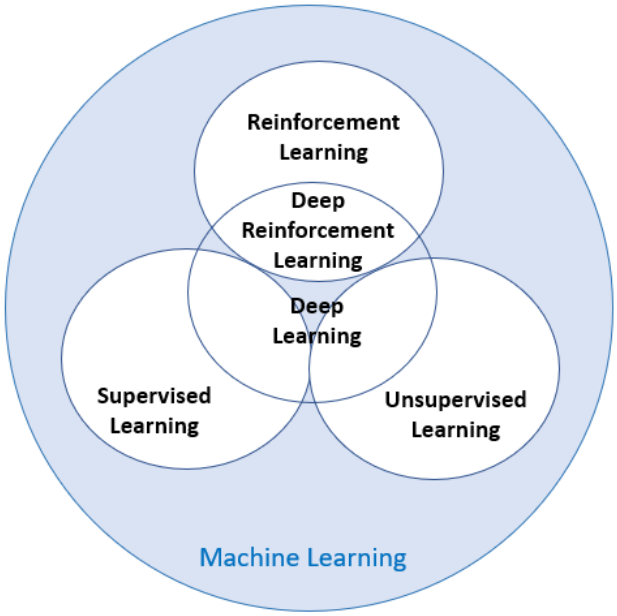

Figure 2.1: *ML learning paradigms based on the type of training.*

Some well-known algorithms following this paradigm are linear and logistic regression [53], support vector machine (SVM) [54], K-nearest neighbors (KNN) [55], and artificial neural networks (ANN) [56].

The main advantage of supervised learning is that, given a good dataset, it can make very good predictions after training. They can also be further enhanced due to their ability to undergo re-training with the introduction of new labeled data, contributing to sustained and improved performance over time. Another advantage is that the labeled data helps define a clear objective for the model, as it learns to map input features to specific output labels. They are also versatile and applicable across various domains. However, acquiring a labeled dataset can be hard and expensive. If the dataset is biased or not representative of the real-world scenario, the model will likely perpetuate those biases, leading to unfair or inaccurate predictions. In general, supervised models are very vulnerable to the bias of those responsible for labeling [57] [58], and they should ensure the representativeness and impartiality of the training data.

Applying supervised learning to optimize the CW is impractical because the network conditions are always changing, and labeling data with the correct CW values would require an understanding of most of the network conditions and scenarios.

2.1.2 Unsupervised Learning

In the unsupervised learning paradigm, the ML model learns to find (sometimes hidden) useful patterns by exploring the input data without labels (i.e., without ex-

pected output values) [59]. The model is trained to create a small representation of the data [52]. This learning paradigm includes the following algorithms: K-means [60], isolation forest [61], hierarchical clustering [62], and expectation–maximization [63]. The main advantage of unsupervised learning is that since it does not rely on labeled data, it is good at identifying hidden patterns [64], which can be useful when analyzing raw data. However, the training phase can be time-consuming, especially with large datasets or complex structures. Since there is no labeled data, it can be challenging to evaluate and validate the results.

Applying unsupervised learning to optimize the CW is not practical because it requires limited control over the learning process and lacks a feedback loop for ongoing adjustments and optimizations.

2.1.3 Reinforcement Learning

A single-agent defines this learning paradigm, where in this context is the ML model (i.e., agent) that 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 [65]. 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 [52]. This paradigm includes the following algorithms: Q-learning [66], Deep Q-learning [67], policy gradient learning [68], deep deterministic policy gradient [69], and the multi-armed bandit [70].

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. Moreover, to illustrate the environment, a well-known statistical model called the Markov chain is introduced as the theoretical base for modeling RL algorithms. The Markov chain is a memoryless stochastic process with finite state space and can fall into a terminal state. Figure 2.2 shows a two-state Markov transition diagram, followed by the index matrix and its transition probability matrix. Furthermore, resembling the Markov chain model, the Markov Decision Process (MDP) is generally adopted to represent the environment of the RL agent because it has a mathematical structure suitable for modeling decision-making problems [52].

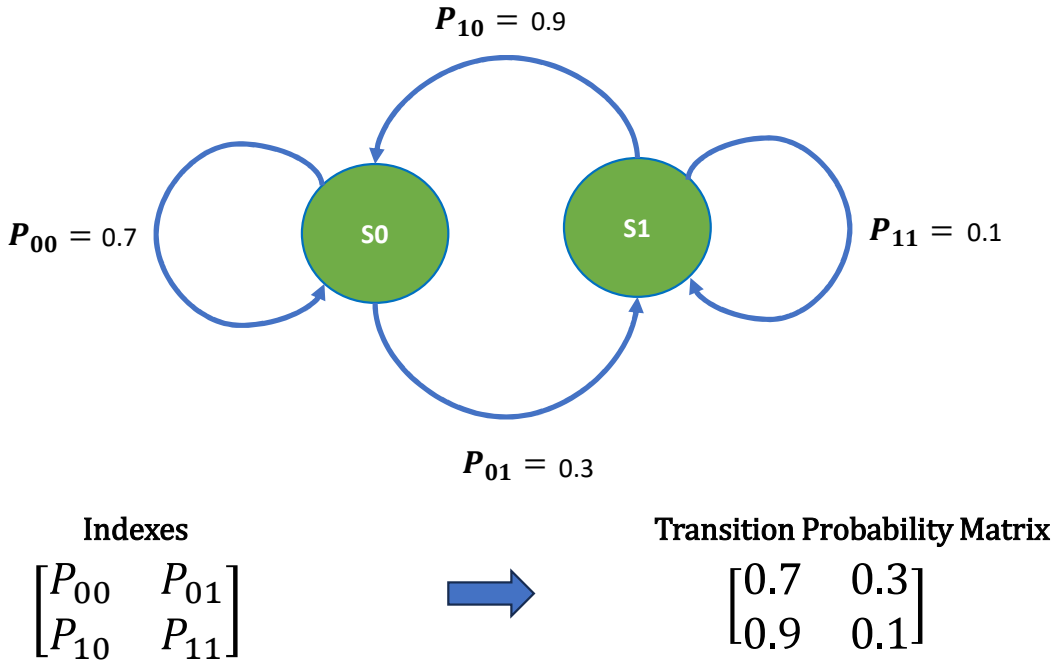

Figure 2.2: *Markov Chain State Transition*

Figure 2.3 In the presented diagram, we have two states, s0, and s1, two actions, a0 and a1, that are represented by the transition from one state to another or to itself, and we also have the rewards associated with these actions. The rewards can be positive or negative. For example, from state s0 to s1 using action a0, we have a reward of 10 values, but from state s1 to itself using action a1 generates a reward of -50, which indicates that this action should be avoided and from state s1 to state s0 has a reward of 40 values, which indicates that this action will be the most chosen by the RL agent to achieve the best total reward. An example of the application of this diagram can be viewed as an electrical game where the player wants to accumulate more points throughput time to win the game. This diagram explains the behavior of how the RL model will learn to make decisions, voiding actions that generate negative rewards while choosing positive ones that maximize the accumulated reward. Figure 2.4 presents the associated matrix related to the MDP state transition diagram with two states followed by the index matrices, the action matrices, and the reward matrices.

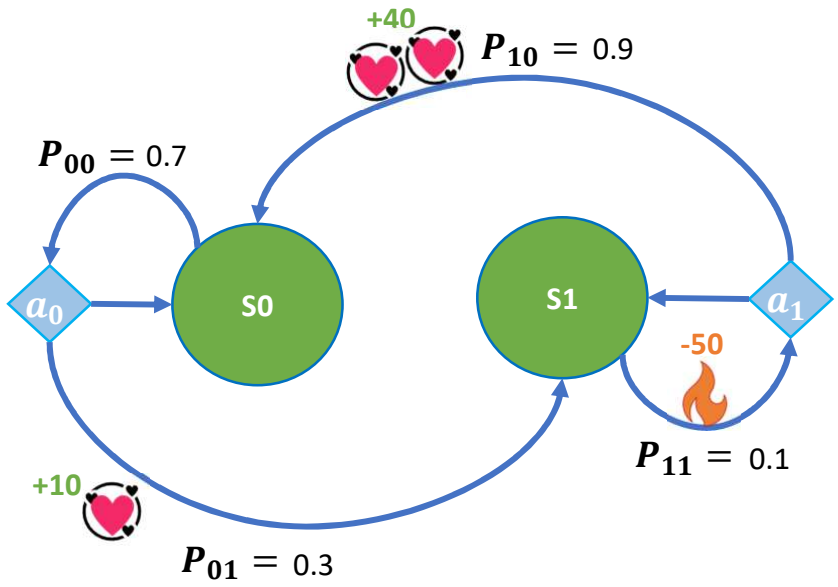

Figure 2.3: *Markov Decision Process State Transition*

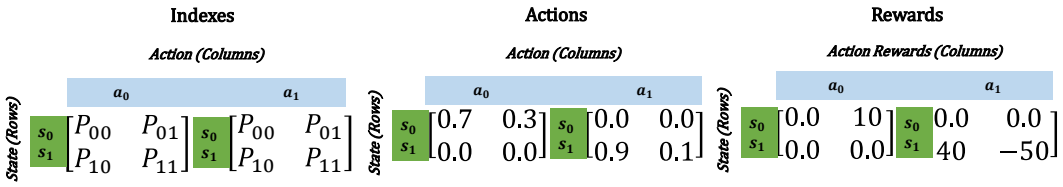

Figure 2.4: *Markov Decision Process State Transition Matrix*

The MDP consists of a tuple of five elements $\mathbf{M} = \{\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \gamma, \boldsymbol{R}\}$, where $\boldsymbol{S}$ is the state space, $\boldsymbol{A}$ is the action space, $\boldsymbol{P}$ is the transition probability, γ is the discount factor, and $\boldsymbol{R}$ is the reward. A reward is a positive or negative numeric value that indicates the quality of the action taken at a particular state [46]. 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 2.5 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.

To better explain the RL learning cycle shown in Figure 2.5, we provide a classic, well-known RL algorithm presented in the literature, that is, the Q-learning, which has its pseudo-algorithm shown in Algorithm 1. Q-learning is a tabular RL method where the training process occurs basically in a table with the rows as the states, the columns

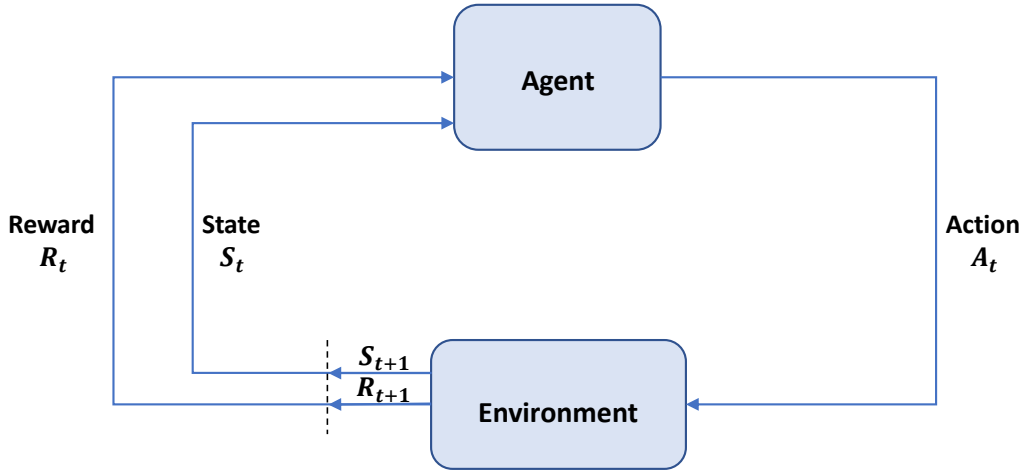

Figure 2.5: *An RL agent interacting with the environment.*

as the actions, and each element inside the table as the Q-value. The goal of Q-learning is to find an optimal Q-value (i.e., measures the quality of the action) that maximizes the reward through iterative updates of the Q-table $Q_t(s, a)$ using the Bellman equation presented in (2.1) [71].

$$Q_{t+1}(s, a) = (1 - \alpha)\left(Q_t(s, a)\right) + \alpha\left[r + \gamma max\left(Q_t(s', a')\right)\right], \tag{2.1}$$

where s is the state, a is the action, r is the reward, α is the learning rate that undertakes an interval of values between $[0, 1]$, γ is the discount rate and uses values between $[0, 1]$, s' is the next-state, a' the next action, and $max(Q_t(s', a'))$ represents the possible maximum reward based on the next-state s' and next-action a'.

A brief description of the Algorithm 1 is provided next. The initialization of the parameters and the Q-table setting occurs from lines 1 through 6. Line 1 defines the learning rate (α). The discount rate is defined in line 2 (γ). Line 3 sets the balance between exploitation and exploration. Then, from lines 7 to 14, the state is initialized, and based on this state, the action is chosen whether to be optimally or randomly selected. The learning phase starts from lines 15 through 21 when the action is applied to the environment being represented by the equation of line 15, producing a new reward, a next-state, and updating the initial parameters, decreasing the exploration rate for each episode.

The basic idea of how Q-learning works after training is shown in Figure 2.6. For each state that is fed into the Q-table, there is a corresponding action, which is the action corresponding to the maximum Q-value for that given input state. In other words, given a line (the state), which action (the column) has the highest Q-value?

Policy: is a rule that helps the agent select the best action in a specific state. The

Algorithm 1 Q-learning

▷ Parameters initialization
Learning rate, α
Discount rate, γ
Exploration rate, ϵ, ϵ_{Min}, ϵ_{Decay}
Initial state, s_0
Action space size, j
Initialize Q-table with zero for all possible (state, actions) pair $Q_t(s, a) \leftarrow 0$

for each Episode **do**
    Reset the state to its initial state $s \leftarrow s_0$

    if random.uniform (0, 1)$<\epsilon$ **then**
        $a \leftarrow$ uniform random integer in [0, j]
    else
        $a \leftarrow$ argmax$(Q_t(s))$
    end if

    After selecting the action a and applying it to the environment, a new reward r and next-state s' are returned from the environment.

    ▷ Learning stage : $Q_{t+1}(s, a)$ is updated with new q-values
    $Q_{t+1}(s, a) \leftarrow (1 - \alpha)(Q_t(s, a)) + \alpha[r + \gamma max(Q_t(s', a'))]$
    $s \leftarrow s'$

    Less exploration of the environment is achieved by decreasing ϵ.
    if $\epsilon >$ ϵ_{Min} **then**
        $\epsilon \leftarrow \epsilon \times \epsilon_{Decay}$
    end if
end for

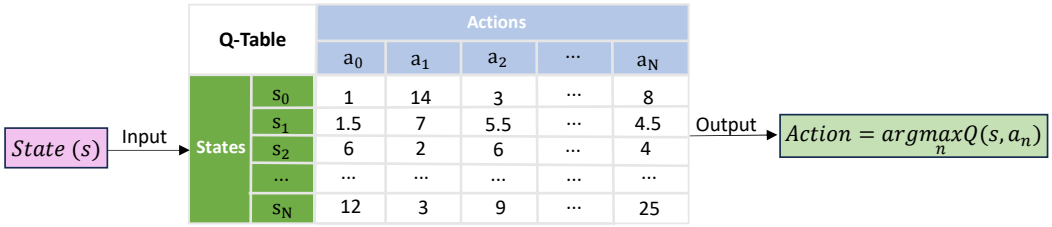

Figure 2.6: *Q-learning.*

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 a trade-off between exploration of the environment

and exploitation of the learned policy [72]. This trade-off can be explained as follows: in the initial phase of training, the RL agent takes random actions to explore the environment with possible actions that might lead to better rewards, since at the beginning, the agent does not know the environment it is in, so the agent is exploring using random actions to understand how the environment works. As training progresses, the agent takes more steps, gaining more knowledge. As the agent gains more knowledge, the random actions decrease, and the agent is most likely to exploit the environment, selecting known actions that provide optimal rewards. An intuitive example of this trade-off is choosing a restaurant for lunch. One has a preference for a specific restaurant knowing that the food there is great and can be considered to be the favorite place to eat (i.e., exploit), but now and then, it is interesting to try something new going to a different restaurant and the taste of the food there might be even better (i.e., explore). So, to maximize the cumulative future reward, the RL agent needs to find a balance between the amount of step time used for exploration (i.e., actions that might return the best rewards) and for exploitation (i.e., actions with the best rewards). 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 [73]. Normally, ϵ is a fixed hyper-parameter, but it can have its value decreased so that the agent explores the environment progressively less. Eq. 2.2 summarizes the ϵ-greedy exploration-exploitation mechanism used by RL algorithms to learn the best policy.

$$\text{Action} = \begin{cases} \text{Random action (Exploration)}, & \text{if random number} < \epsilon, \\ \text{Best long-term action (Exploitation)}, & \text{otherwise.} \end{cases} \tag{2.2}$$

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 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.

2.1.4 Deep Reinforcement Learning

DRL is an improved extension of RL that integrates deep learning (DL), i.e., ANNs, with reinforcement learning algorithms [74]. This integration happens because RL algorithms present limitation problems related to state/action spaces, computational, and

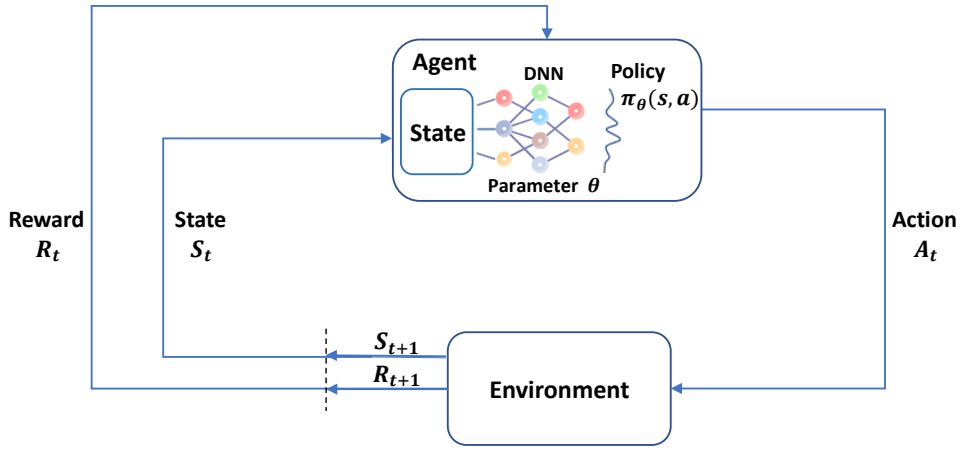

Figure 2.7: *DRL Structure.*

sample complexity [75]. That is, RL algorithms are not scalable and are limited to low-dimensional data issues, i.e., problems with a small number of actions and states [76]. 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 2.7 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 includes the following algorithms: DQN, DDPG, Twin Delayed Deep Deterministic Policy Gradient (TD3), Double Deep Q-learning Network (DDQN), Proximal Policy Optimization (PPO), and Soft Actor-Critic (SAC) [77]. This work focuses on 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.

Deep Q-Network

DQN is an off-policy DRL algorithm based on Q-learning with discrete action space and continuous state space [78]. 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, For example, Q-learning has scaling issues to larger amounts of state and action space for being a tabular method. Therefore, DQN was developed to fix the scaling problem of Q-learning. 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 [79]. This random set of past experiences mitigates the bias that might stem from the fact that some environments have a sequential nature [79]. Here the agent is represented by a

deep neural network (DNN) that uses the state as the input of the neural network, and the outputs are the Q-values corresponding to the possible actions, where the action taken by the agent is that which maximizes the Q-values, as shown in Figure 2.8.

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 [74]. 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 training stability of the predictive network. 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.

Next, is provided a detailed explanation of the DQN algorithm 2, lines 1 to 9, referring to the initialization. The learning rate (α) in line 1 defines the step size to achieve an optimal solution during training. The discount rate in line 2 (γ) sets how much future rewards are taken into consideration [80]. Line 3 sets the balance between exploitation and exploration. Line 4 sets the replay buffer (E), the replay buffer is used to save previous transitions observed by the agent. Line 5 initializes the prediction network (Q^P) with random weights (θ^P). Line 6 initializes the target network (Q^T) with the same weights as the prediction network. Line 7 is the initial environment observation state. In Line 8, the action space size (j) represents the number of actions that the agent can perform. In line 9, the interaction step (K) represents the frequency that the target network updates with the weights of the prediction network. For each episode (line 10), the state is initialized, and the observation sequence is preprocessed (lines 11 and 12). For each step time (line 13), the agent chooses an action, balancing exploration and exploitation (lines 14-18). Then, the selected action is executed in the environment, resulting in a new state and reward (lines 19-21). Line 22 stores the transition tuple in the experience replay buffer. A mini-batch of experiences is sampled from the replay buffer (line 23). Then, the target Q-values are calculated and used to update the parameters of prediction network through gradient descent (lines 24-29). Line 30 is executed at every K interaction step and copies the weights from the prediction network to the target network.

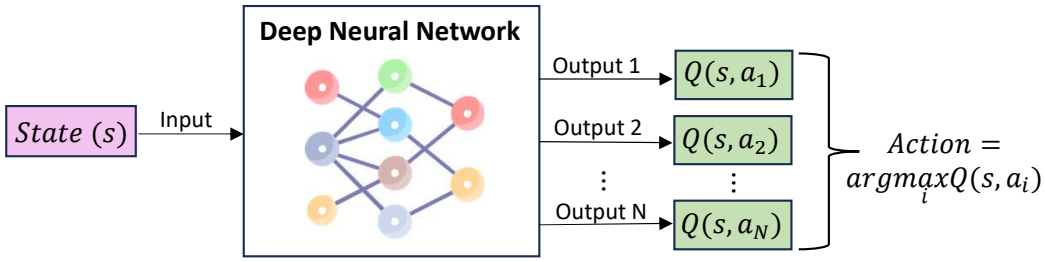

Figure 2.8: *Deep Q-learning (DQN).*

Deep Deterministic Policy Gradient

DDPG is another off-policy DRL algorithm with continuous action and state spaces proposed in [81]. It is the result of the combination of deterministic policy gradient (DPG) and DQN algorithms, the former related to the actor-critic algorithm [39, 82]. 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 [83], where the actor takes actions according to a specific policy function. The critic plays the role of an evaluator of the action taken [39]. Figure 2.9 presents a general view of how DDPG works. Here, the actor takes as input the state and outputs an action. The critic receives as input the state and the action from the actor, which are used to evaluate the actions of the actor, and outputs Q-values corresponding to the set of possible actions. The Q-values outputted by the critic indicate to the agent how good the action taken by the actor for that specific state was [84].

DDPG consists of four networks: *i*) actor prediction, *ii*) critic prediction, *iii*) actor target, and *iv*) critic target networks. The target networks have their weights copied from the prediction networks periodically. The prediction network of the actor is responsible for learning to choose the best action in a given state. The critic prediction network estimates the value of the actions chosen by the actor. The actor target network is updated with the weights that originate from the actor prediction network. The critic target network is updated with the weights that originate from the critic prediction network. Similar to 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 [81].

Similarly to DQN, DDPG uses an experience replay memory to minimize correlations between samples [85]. Regarding the policy aspect of exploration and exploitation, DDPG differs from DQN. Since DDPG works in continuous action space, ex-

Algorithm 2 Deep Q-learning Network (DQN)

▷ ### Parameters initialization ###
Learning rate, α
Discount rate, γ
Exploration rate, ϵ, ϵ_{Min}, ϵ_{Decay}
Initialize the experience replay buffer, *E* with a specific value.
Initialize the prediction network Q^P with random weights θ^P
Initialize the target network Q^T with weights $\theta^T = \theta^P$
Observation sequence, $\{ob_0\}$
Action space size, j
Interaction step, K

for each Episode **do**
    Initial state is initiated with an observation sequence $s \leftarrow \{ob_0\}$
    Preprocess the initial state observation sequence $\phi_1 \leftarrow \phi(s)$
    for each Step Time **do**

▷ ### Pre-learning stage for sampling ###
        if random.uniform (0, 1)$<\epsilon$ **then**
            $a_t \leftarrow$ uniform random integer in [0, j]
        else
            $a_t \leftarrow$ argmax($Q(\phi(s), a; \theta)$
        end if

        After selecting the action a and executing it to the environment, a new reward r and next-state observation ob_{t+1} are returned from the environment.
        Update the next-state observation $s_{t+1} \leftarrow s_t, a_t, ob_{t+1}$
        Preprocess the next-state observation sequence $\phi_{t+1} \leftarrow \phi(s_{t+1})$
        Store the transition in a replay memory $E.append(\phi_t, a_t, r_t, \phi_{t+1})$

▷ ### Learning stage ###
        $mb \leftarrow$ get random mini-batch transitions $(\phi_i, a_i, r_i, \phi_{i+1})$ from E
        if at step i+1 the Episode finishes **then**
            Update the target value to $Y_i \leftarrow r_i$
        else
            Update the target value to $Y_i \leftarrow r_i + max_{a'} Q^T(\phi_{i+1}, a'; \theta^T)$
        end if
        Calculate gradient descent to minimize the loss considering the actual network parameters $\theta \leftarrow (Y_i - Q^P(\phi_i, a_i; \theta))^2$
        At every K interaction step, a copy of the weight of the prediction network is passed on to update the weight of thetarget network.
    end for
end for

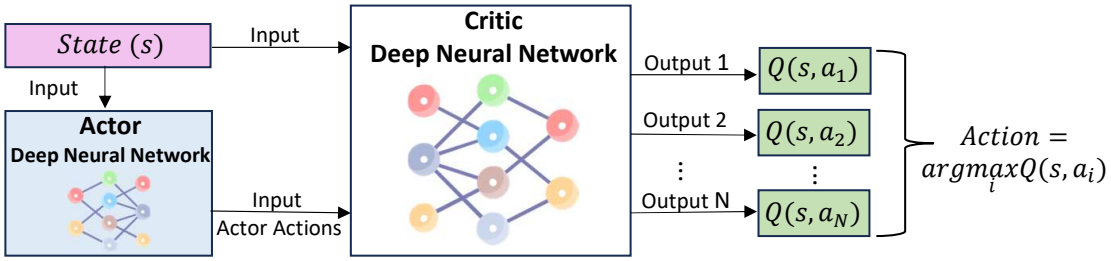

Figure 2.9: *Deep Deterministic Policy Gradient (DDPG).*

ploring 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 [81]. 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 [86].

A detailed description of the DDPG algorithm 3 is provided as follows. Lines 1 to 9 refer to the initialization. Line 1 defines the learning rate (α). The discount rate is defined in line 2 (γ). Line 3 sets the balance between exploitation and exploration. Line 4 sets the replay buffer (E). Lines 5-6 are the initialization of the actor and critic prediction networks with their weights (θ^H and θ^G) and the target network with the same weights. For each episode (line 7), a random noise is initialized to assist the action exploration phase (line 8), and an initial state observation is obtained from the environment (line 9). For each step time (line 10), the agent chooses an action, balancing exploration and exploitation (lines 11-15). Then, the selected action is executed in the environment, resulting in a new state and reward (line 16). Line 17 stores the transition in the experience replay buffer. Afterward, the learning stage starts by sampling a mini-batch of experiences from the replay buffer (line 18). Line 19 updates the target value using the Bellman equation. Line 21 updates the critic value with gradient descent to decrease the loss. In line 23, the policy gradient equation is used to update the actor policy. Lines 25 and 26 update the weight of the network.

Algorithm 3 Deep Deterministic Policy Gradient (DDPG)

▷ ### Parameters initialization ###

Learning rate, α

Discount rate, γ

Exploration rate, ϵ, ϵ_{Min}, ϵ_{Decay}

Initialize the experience replay buffer *E* with a specific value.

Initialize the actor prediction network $H(s|\theta^H)$ and critic prediction network $G(s,a|\theta^G)$ with weights θ^H and θ^G.

Initialize the actor target network $H^{'}$ and critic target network $G^{'}$ with weights $\theta^{H'} \leftarrow \theta^H$ and $\theta^{G'} \leftarrow \theta^G$.

for each Episode **do**

    Initialize a random $Noise$ to assist the action exploration phase.

    Get initial state observation, s

    for each Step Time **do**

▷ ### Pre-learning stage for sampling ###

        if random.uniform (0, 1)<ϵ **then**

            $a_t \leftarrow H(s_t|\theta^H) + Noise$

        else

            $a_t \leftarrow$ argmax($H(s_t, a_t; \theta^H)$

        end if

        After selecting the action a and executing it to the environment, a new reward r and next-state s_{t+1} are returned from the environment.

        Store the transition in a replay memory $E.append(s_t, a_t, r_t, s_{t+1})$

▷ ### Learning stage ###

        $mb \leftarrow$ get random mini-batch transitions (s_j, a_j, r_j, s_{j+1}) from E

        Update the target value to $Y_j \leftarrow r_j + \gamma G^{'}(s_{j+1}, H'(s_{j+1}|\theta^{H'})|\theta^{G'})$

        Calculate and update critic value by using gradient descent to minimize the loss:

$$Loss \leftarrow \frac{1}{mb}\sum_{j=1}^{mb}(Y_j - G(s_j, a_j|\theta^G))^2$$

        The actor policy is updated by utilizing policy gradient:

$$\nabla_{\theta^H} \approx \frac{1}{mb}\sum_{j=1}^{mb}\nabla_a G(s,a|\theta^G)|_{s=s_j, a=H(s_j)}\nabla_{\theta^H} H(s|\theta^H)|_{s_j}$$

        The weight of the target network is updated:

        $\theta^{G'} \leftarrow \tau\theta^G + (1-\tau)\theta^{G'}$

        $\theta^{H'} \leftarrow \tau\theta^H + (1-\tau)\theta^{H'}$

    end for

end for

Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an on-policy DRL algorithm with either discrete or continuous action spaces, designed to improve stability and performance in policy learning. It ensures that policy updates are not too drastic by using a clipping mechanism that limits how much the new policy can change from the old one, that is, it restricts the region from updating the policy by clipping. This approach keeps learning safe and stable while still allowing the policy to improve effectively. The simplicity and effectiveness of PPO make it a popular choice in reinforcement learning, particularly in environments requiring continuous control, such as robotics and gaming.

Soft Actor-Critic

The Soft Actor-Critic (SAC) is an off-policy DRL algorithm with a stochastic actor with continuous action space that improves how policies are learned by focusing on both maximizing rewards and maintaining randomness in decisions. Unlike traditional methods that may become too deterministic early on, SAC encourages the policy to stay flexible, which helps balance exploring new strategies and exploiting known ones. This leads to more stable and efficient learning. SAC uses an actor-critic setup, where one part (the actor) updates the policy based on feedback from another part (the critic) that assesses the value of actions. It also automatically adjusts how much randomness is kept in the policy, this randomness is also referred to as the entropy it measures the randomness of the policy ensuring optimal exploration throughout the training process. This makes SAC a reliable and effective choice for solving complex tasks where controlling continuous actions is required.

2.1.5 Multi-Agent Reinforcement Learning

In this section, we present the theoretical basis of a decentralized solution for collision avoidance in Wi-Fi networks. A multi-agent RL (MARL) is an approach that extends from the single-agent RL for decision-making and optimization problems with multiple self-learning agents sharing the same environment to achieve a certain goal [87]. The theoretical concept of game theory significantly contributes to MARL, where two or more agents aim to maximize their reward individually based on their local information without considering the existence of other agents, and there is no information being shared among the agents, so they act competitively [88]. Each agent in a multi-agent scenario is treated as an independent learner. Therefore, there is no communication exchange between the agents in such a learning process. Independent learning agents cannot access the actions and rewards of others. Furthermore, a rep-

resentation of the multiple-agent interaction with the environment is shown in Figure 2.10.

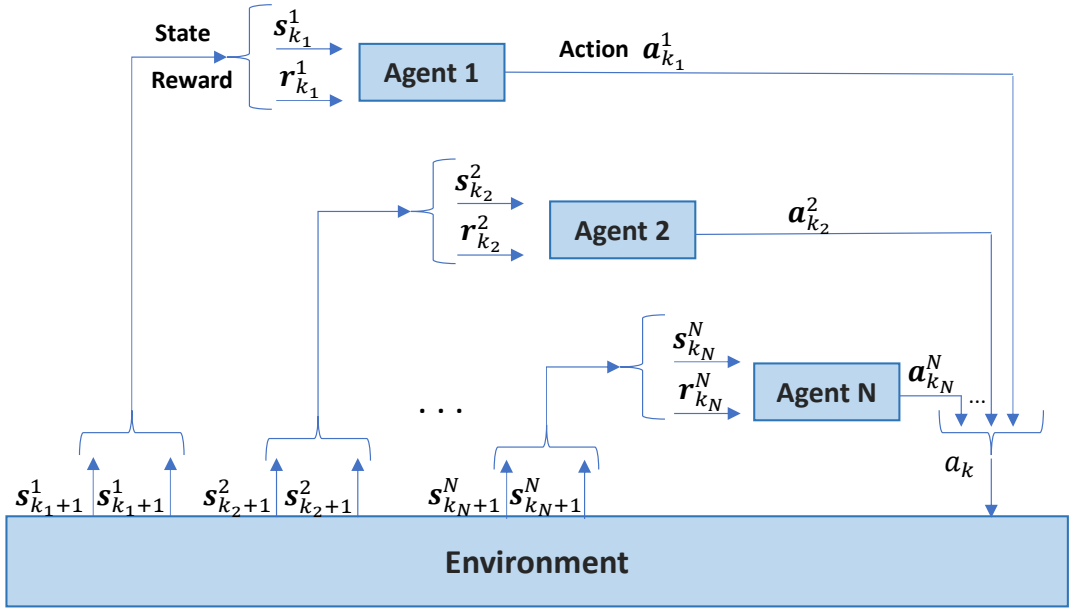

Figure 2.10: *Multi-agent reinforcement learning environment.*

Applying a decentralized approach to WLAN networks offers considerable benefits, including accelerating convergence to optimal network solutions in terms of performance, channel access, collision avoidance, and resource management, for example. By leveraging distributed computing, new agents can be easily added to the corresponding scenario, ensuring a high scalability level, which is a major drawback of single-agent RL. Depending on the training mode and the objective, agents can exchange information, acquiring experience for increased convergence. The multi-agent system is intrinsically robust and its robustness is proven when a sudden failure occurs to some agents in the system, the rest of them continue working without being affected.

Besides its benefits, decentralized MARL presents challenges, including the non-stationarity problem [87] [88]. This issue occurs because each agent in the system is learning concurrently, and the local actions of the agents cause an impact on the transition of the environment, leading to the problem of moving targets. This means that the optimal policy will change whenever the policies of other agents change. The computational complexity grows exponentially as the number of agents being added increases. Another challenging task is regarding coordination (i.e., collaboration) between agents to achieve a greater maximizing cumulative reward. Making agents work collaboratively is difficult because they only have access to their local information, incomplete information about the environment, and no information about the behav-

ior of other agents. Furthermore, multi-agent systems present greater computational complexity, and implementing as well as managing such system is expensive.

2.2 Literature Review

The literature presents significant contributions of ML methods applied to CW optimization in wireless networks. The following contributions implement the ML methods in a centralized manner by applying a single-gent algorithm. For example, in [89], the authors propose a CW optimization mechanism for IEEE 802.11ax under dynamically varying network conditions employing DRL algorithms. The DRL algorithms were implemented on the NS-3 [90] simulator using the NS3-gym [91] framework, which enables integration with python frameworks [91]. They proved to have efficiency close to optimal according to the throughput result that remained stable even when the network topology changed dynamically. Their solution uses the collision probability (i.e., the transmission failure probability), to observe the overall status of the network. Table 2.1 presents a brief summary of the related works regarding the DRL-based centralized approach for CW optimization in wireless networks.

To allow channel access and a fair share of the unlicensed spectrum between wireless nodes, the authors in [92] 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 with decision trees, which are a supervised classification algorithm. Simulations were conducted on a C++ discrete-event-based simulator called CSIM [98] to evaluate the performance of the algorithm. It was possible to obtain high throughput efficiency while maintaining fair allocation of the unlicensed channels with other coexisting nodes.

In [93], the authors present a Deep Q-learning algorithm to adapt CWmin to random access in wireless networks dynamically. The idea is to maximize a network utility function (i.e., a metric measuring the fair use of the medium) [99] 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

Table 2.1: *A summary of the related works using a centralized DRL approach.*

Reference	Wireless Network	Description	Learning Algorithms	Year
[92]	WLANs	Channel access for nodes on the unlicensed spectrum	Random forest with decision trees	2019
[93]	WLANs	CW adaptation for random access on WLANs	Deep Q-Learning	2021
[20]	WLANs	Dynamic CW adjustment for improved network throughput	Fixed-Share	2019
[94]	WLANs	CW optimization for performance enhancement	Genetic fuzzy CW optimization	2021
[95]	MANETs	CW optimization to avoid packet collision	Deep Q-Network	2020
[96]	WLANs	CW optimization to avoid packet collision	Deep Q-Network	2022
[97]	Machine-type communica-tions	Finding better transmission policy for random access	Deep Q-Network	2022

to follow a more complex process. The RL algorithm achieves utility close to optimal when compared to a supervised random forest classifier.

In [20], 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: binary exponential backoff (BEB) and History-Based Adaptive Backoff (HBAB).

To optimize CW in a wireless local area network, [94] 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) [100, 101], the authors of [95] 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 latency. The benefits are even more significant when the queue size is less or equal to 20.

In [96], a different approach to controlling 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 authors of [97] apply DRL to the problem of random access in machine-type communications networks adopting slotted ALOHA access protocols. Their proposed solution aims to find a better transmission policy for slotted ALOHA protocols. The proposed algorithm learns a policy that establishes a trade-off between user fairness and the overall network throughput. The solution employs centralized training, which

makes the learned policy equal for all users. Their approach uses binary feedback signals and past actions to learn transmission probabilities and adapt to traffic scenarios with different arrival processes. Their results show that the proposed algorithm outperforms the classical solution, (i.e., the exponential back-off), in terms of user fairness and overall throughput. Our proposed solution differs from this one because it tackles the problem of random access in slot-free and contention-based networks, aiming to maximize the network throughput and, consequently, the throughput of individual users.

Differently from the previous centralized solutions, in this work, we propose leveraging an alternative observation metric that is based on the average normalized transmission queue levels of all stations, which seems a more informative and straightforward way to understand and capture the overall status of the network, to train DRL agents to take actions aiming at finding the best CW value, which in turn, optimizes the network performance. Moreover, we compare the observation metric proposed in [89] (i.e., the collision probability observed by the network) with the proposed one, (i.e., the averaged normalized transmission queue levels of the stations). The level of normalized transmission queues provides more information, taking into account the congestion level of the network and the behavior of all nodes, not just the individual nodes. It also leads to better performance, allows for adaptation to changing network conditions, and avoids myopic decisions. In contrast, using the collision probability alone can result in suboptimal decisions that do not take into account the impact on the overall network performance. Therefore, the averaged normalized transmission queue levels seem to be a more suitable metric for RL solutions to solve the collision avoidance problem in Wi-Fi networks.

Several multi-agent RL methods have been proposed in the literature to enhance WLANs performance by providing decentralized solutions. Table 2.2 presents a brief summary of the related works regarding the DRL-based decentralized approach for CW optimization in wireless networks.

The authors in [42] present a multi-agent deep learning solution for optimal prediction of the spectrum occupation to improve the multi-channel slotted wireless network access. The wireless spectrum is being overused and becoming a scarce resource due to the fact that many network technologies, as well as electromagnetic sources, coexist in the same radio bands, leading to collisions with the peripheral networks. To mitigate the collision occurrence between the networks, a multi-agent DRL mechanism is proposed, which comprises six known nodes and two unknown nodes following two topologies (ring topology and AP topology). A distributed RL-based scheduler is used for the slot allocation to identify which slot will be used next to avoid collisions. The

Table 2.2: *A summary of the related works using a decentralized DRL approach.*

Reference	**Wireless Network**	**Description**	**Learning Algorithms**	**Year**
[42]	Multi-channel slotted WLANs	Multi-agent spectrum prediction	Deep neural network using online supervised learning	2020
[85]	Heterogeneous WLANs	Selecting the best channel bonding parameters for access points	Multi-agent deep deterministic policy gradient	2020
[102]	Multi-cell OFDM	Determines the power control and joint subcarrier allocation problem	Distributed Q-Learning	2020
[103]	HetNet	Power allocation for improved spectral access	Multiple-actor-shared-critic	2020

agent training is conducted by applying online supervised learning with experience replay. The radio frequency (RF) observation is the information that each node uses to make its prediction and run its learning procedure. Based on the proposed solution was possible to have a 30% decrease in inter-network collisions compared to the conventional schedulers. In terms of the overall throughput, the improvement of the proposed algorithm compared to the traditional exponentially weighted moving average (EWMA) algorithm had an increase of 10% in a network with a medium size.

In [85], the authors propose an on-demand channel bonding (O-DCB) algorithm based on DRL for heterogeneous WLANs. In a wireless network comprising multiple WLANs with varying channel bonding capabilities, interference among WLANs is possible, while each AP experiences different traffic demands. The goal of the proposed work is to determine the best channel bonding parameters for all APs, considering their bonding capabilities, fluctuating traffic demands, and potential interference, reducing the average transmission delay across the entire network. This delay is defined as the time elapsed between a packet arriving at the AP buffer and the packet being successfully received. The problem involves continuous state space and discrete action space, with the action space growing exponentially with AP numbers. To speed up the learning process, a multi-agent deep deterministic policy gradient (MADDPG) is employed to train O-DCB. Real traffic traces from a campus WLAN validate the proposed algorithm, showing good convergence and lower delay compared to other methods.

To optimize the system spectral efficiency, the authors in [102] propose a MARL algorithm to solve the problem of power allocation and joint subcarrier in a multi-cell multi-cell orthogonal frequency-division multiplexing (OFDM) system. With this MARL algorithm, each base station (BS) in this approach performs resource allocation calculations independently based on the conditions of the wireless environment. However, these BSs act as agents capable of information exchange, enabling collaborative calculation for global resource allocation. The presented numerical results demonstrate that the proposed MARL algorithm has fast convergence and can reach an efficiency of up to 53.6% compared to the conventional Q-learning RL algorithm.

The multiplicity of APs and users in a heterogeneous network (HetNet) sharing the same spectrum band can lead to significant interference issues. To effectively manage this interference and improve network performance, advanced power control techniques become imperative. Generally, power control techniques rely on gathering instantaneous global channel state information (CSI) and then calculating sub-optimal solutions. However, in a HetNet environment, obtaining instantaneous global CSI is challenging due to its fast-changing nature. To address this challenge, the authors

of [103] propose a novel approach using DRL to design a multi-agent power control algorithm. Each AP is treated as an agent equipped with a local DNN. The proposed method, called multiple-actor-shared-critic (MASC), trains these local DNNs separately in an online trial-and-error manner. Each AP can independently use its local DNN to control transmit power based on only local observations. Simulation results demonstrate that the proposed algorithm surpasses conventional power control algorithms in terms of both converged average sum rate and computational complexity.

The decentralized solution proposed in this work makes a fine distinction among the previously mentioned works. We aim to evaluate the hypothesis of how well a decentralized approach can effectively optimize the CW value in different scenarios to achieve optimal network performance using the collision probability as the observation metric for training the DRL agents by taking actions seeking the greatest cumulative reward. Additionally, a comparison is made considering the collision probability observed by the network of the centralized and decentralized approach over the conventional BEB algorithm. It was proven that the decentralized results lead to better performance over BEB, allowing for adaptation to changing conditions of the network. For this reason, the decentralized approach is apt for handling the collision avoidance issue in WLANs.

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 [104]. Therefore, this motivated us to study and propose a DRL-based solution to reduce collisions by optimizing CW in different scenarios.

Chapter 3

Proposed Methodology

IN order to apply DRL algorithms (DQN and DDPG) to optimize Wi-Fi networks, we propose a centralized and decentralized approach for solving the CW optimization in this work. For the centralized case, 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 performance (i.e., the reward). Unlike previous centralized approaches, this work introduces a different observation metric based on the average normalized transmission queue levels of all stations. This metric provides more detailed information to analyze and evaluate the overall network status, used to train DRL agents to determine the optimal CW value, enhancing network performance. Additionally, we compare the proposed average normalized transmission queue levels with the observed collision probability within the network presented in [89]. For the decentralized case, the decentralized algorithm will be the module running in multiple stations simultaneously observing the state of the network and choosing the appropriate CW values for each station separately to optimize the performance of the network. The decentralized solution proposed in this work, differs from the previous ones by treating 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. The proposed decentralized solution uses collision probability as the observation metric for DRL agents. Furthermore, we compare the collision probabilities of the centralized versus decentralized approach over the conventional BEB algorithm. Next, we present some details on the agent and its inputs and output values for both centralized and decentralized approaches.

1. **Agent:** the agent represents the proposed DRL algorithms (i.e., DQN and DDPG). The agent is chosen to run on the AP since it has a general view of the whole net-

work and then can control the stations associated with it through beacon frames in a centralized way. Therefore, as can be noticed, this is a centralized approach where the AP decides the best CW value that will be used across the network. For the decentralized approach the DRL agents (i.e., DQN and DDPG) run on multiple stations varying from 5 to 50 having each station selecting its appropriate CW value.

2. **Current state** (s): the current state 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 optimization problem. Therefore, we model the problem as a Partially Observable Markov Decision Process (POMDP) instead of an MDP one. POMDP assumes the state of the environment cannot be perfectly observed [105].

3. **Observation** (o): is the acquired information from the network based on the averaged normalized transmission queues level of all associated stations or on the probability of collision proposed in [89]. We will employ and compare the performance attained by these two different types of information obtained from the network. Section 3.0.3 explains how each one of these observation values is calculated.

4. **Action** (a): the action corresponds to the CW value. Since the CW value is directly connected to the network performance, longer back-off periods lead to longer waiting times for retransmitting packets in case of collisions, degrading the spectrum usage and the network performance. The RL scheme brings the idea that for every action, there is a related maximized reward. Thus, applying RL concepts to optimize the CW value aiming to maximize the throughput of the network is what this work proposes. Therefore, we use the CW value as the RL action. 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. The set of actions follows these specific values (i.e., 0 through 6) to define the back-off interval used by IEEE 802.11a for a station to retransmit its packet, establishing the limit condition of retransmission retries. Therefore, the CW value to be broadcast to the stations can be obtained through the application of (3.1). This interval is selected so that the action space is within the 802.11 standard 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, $T(s'|s, a)$. Note that the action, a, is mapped into the CW value by using equation (3.1). When using (3.1), the CW interval, which is broadcast by

the AP, is in the range of 15 to 1023, according to the 802.11 standard.

$$CW = \left\lfloor 2^{a+4} \right\rfloor - 1. \tag{3.1}$$

5. **Reward** (r): the reward is defined as the normalized network throughput, which can be observed at the AP by the agent 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 its expected maximum.

3.0.1 Problem formulation

The collision occurrence problem in Wi-Fi networks is a topic of great concern, especially when dealing with dense scenarios where multiple stations are trying to access the medium. Especially because traditional collision avoidance mechanisms are not able to optimize the radio resources of the spectrum to efficiently improve the throughput of the network while guaranteeing a fair share of the spectrum between the multiple stations, for this reason, we present two RL-based solutions following a centralized and decentralized approach to avoid collisions in the IEEE 802.11 networks. The centralized approach optimizes the CW value by using only one (i.e., central) RL agent at the AP interacting with the environment. The CW value selected by the RL agent from the AP will be distributed among all the stations, which means that each station will have the same CW value. To train the RL agent, we used two types of observation metrics, namely, averaged transmission queues level and the collision probability, to compare the performance of the two observation types. In the decentralized approach, the CW optimization is done by multiple stations independently interacting with the same environment and having separate rewards to maximize the overall reward, that is, each station will have its own RL agent able to select CW values for each of them individually. We use the collision probability for the decentralized solution to train the RL agent and compare centralized and decentralized results. For both approaches, the input of the neural network of the RL agent is the mean and variance of the observation metric. Furthermore, both approaches were simulated for static and dynamic scenarios.

Centralized solution

Typically, a centralized ML system is characterized by a single-agent RL that interacts with the environment and decides which action to take through a process of trial and error and also based on experience. Every time an action is taken in the environ-

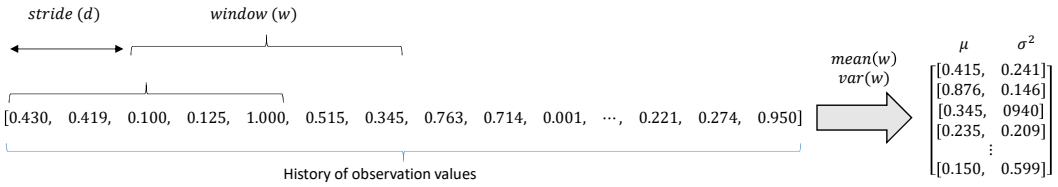

Figure 3.1: *Bi-dimensional observation vector used in the preprocessing phase.*

ment, a new reward, whether it is positive or negative, is generated with the goal of maximizing the cumulative reward. In essence, the reward serves to assess the efficacy of the action taken. Consequently, the agent will endeavor to avoid any negative reward, thereby indicating that such actions are to be evaded to maximize the overall reward. The agent learns by continuously interacting with the environment, and this process is shown in section 2.1.3 with more details in Figure 2.5.

As referred to in section 2.1.3, RL algorithms have the capacity to learn from network states and adjust to evolving network conditions, thereby optimizing cumulative rewards over time. This adaptability renders RL particularly suitable for solving many optimization and decision-making problems of Wi-Fi networks providing optimal solutions, being flexible, and adaptable to different learning scenarios. By contrast, RL-based algorithms remain complex, demanding high computational resources. Conventional single-agent RL algorithms like Q-learning require a stable environment to converge to an optimal solution. However, in dynamic and unstable environments, this stability is compromised, leading to increased computational resource consumption, and the training process may extend over several hours. Additionally, they present scalability issues in managing continuous and high-dimensional state and action spaces.

The proposed approach for the centralized solution has three stages. The first is a pre-learning stage, where the legacy Wi-Fi contention-based mechanism manages the network. This stage is used to initialize the observation buffer with information that will be used to train 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 4.

The mean, μ, and variance, σ^2, of the history of recent observation values (either the averaged normalized queues level or the collision probability) 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 as illustrated in Figure 3.1.

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 addi-

tion 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 the case of DDPG, 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 period of training stage by setting the variable $trainingPeriod$. At this stage, the noisy factor is set to zero, 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 replay memory (called in this work as the experience replay buffer, E), a size limit has to be configured for this memory. The replay 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).

A brief description of the pseudo-code shown in Algorithm 4 is provided next. The initialization phase goes from lines 1 to 15. It initiates the input parameters such as observation and replay buffers, agent weights, initial state, noisy factor, CW value, and variables used to select the current stage and the type of observation employed.

Then, the pre-learning stage from lines 19 to 28 starts by filling in the observation buffer with the selected type of observation metric (either the averaged normalized queues level or the collision probability). The flag $useQueueLevelFlag$ sets the type of observation value to be calculated. In this stage, the observation value (either the averaged normalized transmission queues level or the collision probability of the network) results from the application of the legacy Wi-Fi contention-based mechanism (i.e., BEB). After $envStepTime$, which is the period between interactions of the proposed algorithm with the environment, has elapsed, the algorithm selects the CW value, and then, the observation value results from the application of the DRL-based CW optimization algorithm. Therefore, one should note that the instructions in lines 19 to 28

are shared between the pre-learning and learning stages.

Next, from lines 30 to 33, the information in the observation buffer is pre-processed by applying a moving average operation to the data, which results in a two-dimensional vector, $observation$, with the mean, μ, and variance, σ^2 of the data in the buffer. With the pre-processed data, the action function, A_θ, returns the action, a. The action is then used to determine a new CW value that will be broadcast to all associated stations. The instructions in lines 30 to 33 are shared between the learning and operational stages. The difference between these two stages is how the action, a, is selected. In the learning stage, the $trainingFlag$ is equal to True (line 34), which tells the action function, A_θ, to choose actions following the exploration-exploitation approach described previously. In the operational stage, the $trainingFlag$ is equal to False, which tells the action function, A_θ, to choose actions optimizing the performance of the network. In the learning stage, a noisy factor is used to explore the environment, which does not happen in the operational stage.

After that, from lines 35 to 42, the algorithm is solely in the learning stage. In this section of the algorithm, the DRL agent learns from previous experiences; that is, it has its weights, θ, updated by using mini-batches of samples randomly picked from the replay buffer, E. These mini-batches are composed of the current μ, σ^2, action a values, the reward (i.e., the normalized throughput), r, and the previous values of μ and σ^2.

Finally, in lines 46 to 47, the algorithm checks if the training period has elapsed. If so, the $trainingFlag$ is set to False, and the algorithm enters the operational stage. In this stage, the noisy factor is disabled, and the algorithm only exploits the acquired knowledge.

Decentralized solution

A decentralized reinforcement learning (RL) system involves multiple self-learning agents sharing an environment to achieve a common objective without communicating or knowing about one another. Based on game theory, these agents act independently, maximizing their rewards according to local information. This approach applied to WLAN networks, is known to enhance performance and scalability while ensuring robustness against agent failures since agents are deployed in a distributed manner. However, challenges include non-stationarity, where changing policies of the agents affect the environment, increasing computational complexity with more agents, and difficulties in making the agents collaborate due to limited information.

For the decentralized solution, we have the following mapping, one AP and several

Algorithm 4 DRL-based CW Centralized Optimization

▷ ### Initialization ###
Initialize the observation buffer, O, with zeroes
Initialize the weights, θ, of the agent
Get the action function, A_θ, which the agent uses to choose the action according to the current stage
Initialize the interaction period of the algorithm 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, *E*, with zeroes.
trainingStartTime $\leftarrow$ *currentTime*
lastUpdate $\leftarrow$ *currentTime*
$\mu_{prev} \leftarrow 0$ (previous mean value)
$\sigma^2_{prev} \leftarrow 0$ (previous variance value)
Set $useQueueLevelFlag \leftarrow True$ to use the averaged normalized transmission queues level as observation.
CW $\leftarrow 15$

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

▷ ### Pre-learning stage ###
        if $useQueueLevelFlag$ == $True$ **then**
            Get $Q^{(i)}_{NL}$ received from each associated station
            Calculate $\bar{Q}_{NL}$
            observation $\leftarrow \bar{Q}_{NL}$
        else
            $N_t \leftarrow$ get number of transmitted frames
            $N_r \leftarrow$ get number of received frames
            observation $\leftarrow \frac{N_t - N_r}{N_t}$
        end if
        O.append(*observation*)

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

▷ ### Learning and operational stages ###
            $\mu, \sigma^2 \leftarrow preprocess(O)$
            $a \leftarrow A_\theta(\mu, \sigma^2, trainingFlag)$
            $CW \leftarrow 2^{a+4} - 1$
            Broadcast the new CW value to all associated stations

            if $trainingFlag$ == $True$ **then**
                $N_{RP} \leftarrow$ get the number of received packets.
                $tput \leftarrow \frac{N_{RP}}{envStepTime}$
                $r \leftarrow normalize(tput)$
                E.append((μ, σ^2, *a*, *r*, μ_{prev}, σ^2_{prev}))
                $\mu_{prev} \leftarrow \mu$
                $\sigma^2_{prev} \leftarrow \sigma^2$
                $mb \leftarrow$ get random mini-batch from *E*
                Update θ based on *mb*
            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

stations, where the number of stations varies from 5 to 50. Each station will be an independent learning agent using its local collision probability value as the observation metric to train the RL agent. Different from the centralized solution, where the cumulative reward is generated based on the training of only one agent, in the decentralized approach, each agent is individually rewarded, and the cumulative reward will be the total sum of the individual rewards of the stations.

The decentralized algorithm approach follows the same three stages as the centralized solution but with some particularities to train the RL agent. A brief description of the pseudo-code is shown in Algorithm 5, with the possibility of training different agents in a distributed manner.

Lines 1 to 15 characterize the initialization phase. During this phase, various input parameters are set up, including the individual observation of the agents, replay buffers, agent weights, initial state, noisy factor, CW value, and variables used for selecting the current stage.

Next, the pre-learning stage occurs from lines 20 to 23, where the observation buffer for each agent starts to be filled with the individually calculated collision probability as the observation metric. This observation buffer will store the observations collected from each station separately in different indexes (i.e., list). During this stage, the observation value results from the application of the legacy Wi-Fi contention-based mechanism(i.e., BEB). After $envStepTime$, which is the period between the proposed interactions of the algorithm with the environment, has elapsed, the algorithm selects the CW value of each station, and then the observation value results from the application of the DRL-based CW optimization algorithm. Therefore, one should note that the instructions in lines 20 to 23 are shared between the pre-learning and learning stages.

Next, from lines 25 to 27, the information in the observation buffer is pre-processed by applying a moving average operation to the data, which results in a two-dimensional vector, $observation$, with the mean, μ, and variance, σ^2 of the data in the buffer. With the pre-processed data of each station, the action function, A_θ, returns individual unique actions for each station, a. The action of each station is used to determine a new CW value for each station. The instructions in lines 25 to 27 are shared between the learning and operational stages. The difference between these two stages is how the action, a, is selected. In the learning stage, the $trainingFlag$ is equal to True (line 28), which tells the action function, A_θ, to choose actions following the exploration-exploitation approach described previously. During this stage, the returned reward (i.e., throughput) is generated by each station after each action taken simultaneously by independent self-learning stations. Each station has its own action that has an impact on

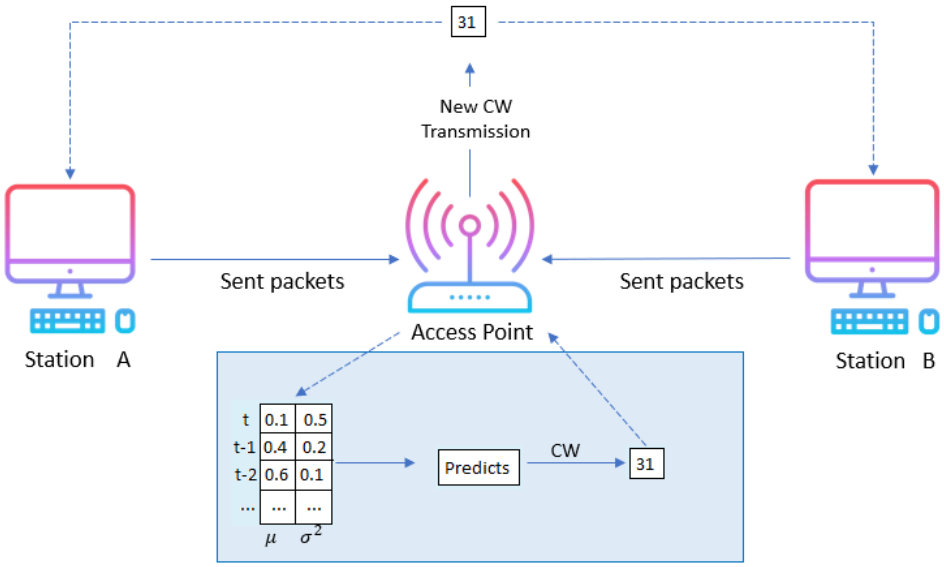

Figure 3.2: *Scenario for assessing the proposed centralized DRL-based CW optimization method.*

the CW value selection, so each action and CW selection from the station generates a reward in the form of throughput. This means that each action taken by each station will produce a new reward, and the reward generated by each station will be sent to the AP, the reward of each station will be accumulated at the AP and will be broadcast to all associated stations to maximize the final reward (i.e., throughput). In the operational stage, the $trainingFlag$ is equal to True, which tells the action function, A_theta, to choose actions that optimize the performance of the network. In the learning stage, a noisy factor is used to explore the environment, which does not happen in the operational stage.

Lines 29 to 38 characterize the learning stage. In this section of the algorithm, the DRL agent learns from previous experiences; that is, it has its weights, θ, updated by using mini-batches of samples randomly picked from the replay buffer, E. These mini-batches are composed of the current μ, σ^2, action a values, the reward (i.e., the normalized throughput) r, and the previous values of μ and σ^2.

Lastly, in lines 42 to 43, the algorithm checks if the training period has elapsed. If so, the $trainingFlag$ is set to False, and the algorithm enters the operational stage. In this stage, the noisy factor is disabled, and the algorithm only exploits the acquired knowledge.

Algorithm 5 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 interaction period of the algorithm 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 $e = 1, ..., N_{episodes}$ **do**
    Reset and run the environment, i.e., reset and run the NS-3 simulator
    for $t = 1, ..., N_{spe}$ **do**
        for $i = 1, ...,$ 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.
                    $tput(i) \leftarrow \frac{N_{RP}}{envStepTime}$
                    Send the throughput of each station to the access point.
                    $r \leftarrow normalize(tput(i))$
                    Broadcast the new r reward value 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

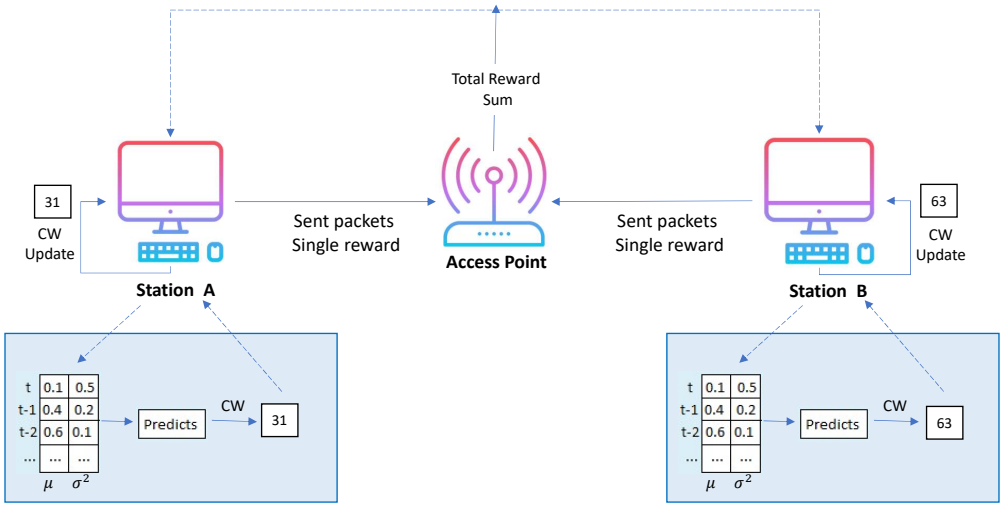

Figure 3.3: *Scenario for assessing the decentralized DRL-based CW optimization method.*

3.0.2 Network topology

We consider a linear topology comprised of one AP and several stations (i.e., going from 5 to 50) transmitting packets. 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 static, where the number of stations is set as a fixed value in the simulation environment, and in the dynamic scenario, the stations are added during the simulation execution. The system model considered in the centralized solution is depicted in Figure 3.2. The AP plays the role of the RL agent, selecting a new CW value according to the current observation, the reward is calculated in a centralized manner. For the decentralized solution, since the stations are autonomous agents, each agent will have its CW value adjusted and receive individual rewards. The cumulative reward is then calculated as the total sum of rewards across all individual stations. In the adopted topology, the stations with RL agents follow a distributed-like topology with one AP is depicted in Figure 3.3.

3.0.3 Observation Metrics

Since the metric characterizing the environment should provide the best possible understanding of the current status of the network, we propose the usage of the averaged transmission queues level and Collision Probability as observation metrics.

Averaged transmission queues level

The averaged normalized transmission queues level of all associated stations, $\bar{Q}_{NL}$, as observation. This metric is adopted as observation because it offers a more direct

way of obtaining information about the overall network status. Next, we describe how it is calculated. Conversely, as will become clear next, the collision probability requires the AP to determine the number of transmitted and correctly received frames, which is, in turn, determined based on the number of transmitted or received acknowledgment frames.

The normalized level of the transmission queue of the i-th station, $Q_{NL}^{(i)}$, is calculated according to (3.2), where (i) indicates the station number. The queue level, $Q_l^{(i)}$, is normalized (i.e., divided) by the maximum queue size value of that station, $Q_{max}^{(i)}$.

$$Q_{NL}^{(i)} = \frac{Q_l^{(i)}}{Q_{max}^{(i)}}. \tag{3.2}$$

The measurement of $Q_{NL}^{(i)}$ is carried out at predefined intervals at each station and indicates the result of the currently chosen CW value on the performance of the network. For example, a value close to 1 indicates the queue is full, meaning the station cannot transmit packets as quickly as it receives them from the upper layers. 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 $Q_{NL}^{(i)}$ 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, $Q_{NL}^{(i)}$, is concatenated (i.e., piggybacked) 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 $Q_{NL}^{(i)}$ coming from the stations by the total number of stations associated with the AP, $N_{stations}$, as shown in (3.3). This is the observation used by the agent to gather insights into the status of the network.

$$\bar{Q}_{NL} = \frac{1}{N_{stations}} \sum_{i=1}^{N_{stations}} Q_{NL}^{(i)}. \tag{3.3}$$

Collision Probability

The collision probability metric is a good characterization of the environment, proposed in [89], is the probability of collision, p_{col}, observed by the network. It can also be interpreted as the probability of transmission failure. This probability is calculated based on the number of transmitted, N_t, and correctly received, N_r, frames, as shown in (3.4).

$$p_{col} = \frac{N_t - N_r}{N_t}. \tag{3.4}$$

This collision rate approximates the actual probability of collision as the number of frames used to calculate it increases. Thus, this rate represents the probability of

a frame not being received due to another station transmitting a frame at the same time. These probabilities are calculated within the interaction periods and provide information on the performance of the selected CW value.

3.0.4 Reward Function

In this work, we assess two metrics: the overall throughput of the network and the CW value. The overall throughput of the network measures how much data is successfully transmitted over a period over the network. The CW value is a random parameter that determines how long a station must wait before it can transmit data. Therefore, one can see that a longer CW value means that stations must wait longer before transmitting, which can lead to lower throughput. However, a longer CW value means a lower chance of collisions. On the other hand, the smaller the CW value, the shorter the waiting time and, consequently, the higher the chance of collisions, which is detrimental to the overall throughput of the network.

The DRL agent employs the CW values as actions and the average of the normalized level of the transmission queues of the stations or the collision rate as observations. Its reward function is the normalized overall throughput of the network. The objective of the agent is maximizing its received reward over time based on observations of the environment. Therefore, the connection of the reward with the throughput is straightforward, (i.e., the higher the reward, the higher the overall throughput of the network).

The connection of the reward with the CW value is also direct if we understand that the longer the CW value, the lower the throughput. It happens because a station cannot transmit during the CW wait time, penalizing its throughput since it is calculated during a given period. On the other hand, the higher the CW value, the lower the chance of collisions, and vice versa. Therefore, the DRL agent aims to find a CW value that maximizes the overall throughput of the network while minimizing frame collisions (i.e., a trade-off between throughput and collisions).

Chapter 4

Results and Analysis

This chapter 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. Our study considers and compares the averaged normalized transmission queue levels of all stations and the probability of collision as two different types of observation metrics to train the DRL-based CW optimization agent. The simulation results presented in this section demonstrate the performance achieved by the DRL-based solution employing either one of the observations in the previously mentioned scenarios.

4.0.1 Simulation setup

The proposed centralized and decentralized DRL-based CW optimization solutions are implemented on NS3-gym [91], which runs on top of the NS-3 simulator [90]. NS3-gym enables the communication between NS-3 (C++) and OpenAI gym framework (python) [106]. 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 [90]. In this work, we use version NS-3.29 of the NS-3 simulator. The DRL algorithms used here were implemented with TensorFlow (version 1.14.0 and PyTorch (version 0.4.1). The simulations were performed on a desktop with an Intel Xeon E5-1620 v3 processor, 32 GB RAM running Ubuntu 20.04.

Table 4.1 presents the NS-3 parameters necessary to create the environment in which the agent will learn. Apart from those parameters, we assume single-user trans-

Table 4.1: *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]
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
Mobility model	ConstantPositionMobilityModel
Simulation time	60 [s]

missions, a packet load adjusted to saturate the network with constant bit rate (CBR) user datagram protocol (UDP) traffic of 150 Mbps, instant and faultless transference of network information, e.g., the normalized queue level of each station, $Q_{NL}^{(i)}$, or the number of transmitted packets, N_t, of each station, 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 $Q_{NL}^{(i)}$ or N_t 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., $Q_{NL}^{(i)}$ or N_t and CW. In this experiment, the station nodes in NS-3 were created using two C++ classes to represent both static and dynamic scenarios. By default, NS-3 uses a mobility model helper to position the generated station and AP nodes according to a 3D Cartesian coordinate system. A list-based position allocator was used to assign coordinates, with the AP node placed at (0.0, 0.0, 0.0) and the station node at (1.0, 0.0, 0.0). The mobility model was set to ConstantPositionMobilityModel, ensuring that the nodes remained stationary throughout the simulation.

Table 4.2 presents the agent parameters used in NS3-gym for the experiments. These agent parameters, including learning and reward discount rates, were ideally

Table 4.2: *NS3-gym agent configuration parameters.*

Configuration Parameter	Value
DQN learning rate	4x10^{-4}
DDPG actor learning rate	4x10^{-4}
DDPG critic learning rate	4x10^{-3}
Reward discount rate	0.7
Batch size	32
Replay memory size	18,000
Size of observation history memory	300
$trainingPeriod$	840 [s]
$envStepTime$ (i.e., interaction interval)	10 [ms]

chosen using a grid search from previous studies in the literature. The network architecture used by both DRL algorithms has one recurrent long short-term memory (LSTM) layer and two fully connected layers leading to an $8 \times 128 \times 64$ topology. An illustration of the network architecture is shown in Figure 4.1.

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 14 executions were part of the training stage (thus, the training stage period, $trainingPeriod$, is equal to $14 \times 60[s] = 840[s]$). The last one was the operational stage. Each one of the simulations consisted of 10 [ms] interaction intervals (i.e., $envStepTime$). Algorithm 4 and 5 were executed between these interaction intervals.

4.0.2 Centralized Solution Results

Static scenario results

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 the IEEE 802.11 conventional BEB algorithm.

Figure 4.2 shows the throughput achieved by the network for different numbers of stations. As can be seen, the network throughput decreases as the number of stations

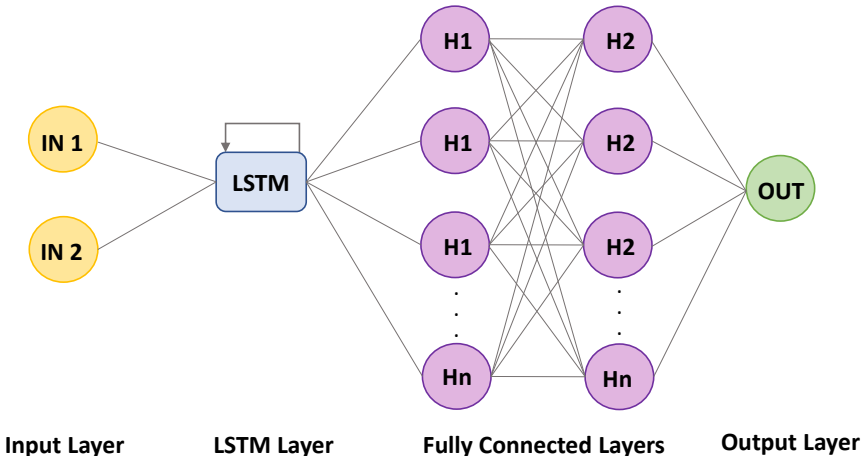

Figure 4.1: *Architecture of the Deep Learning Network.*

increases when BEB is employed. On the other hand, when either DQN or DDPG is used with either one of the observation metrics, it remains practically constant as the number of stations increases, proving the hypothesis. In this scenario, the graph shows that the throughput values achieved by our DRL-based solution with four different numbers of stations (5, 15, 30, and 50) are very similar when using either one of the metrics. This is indicated by the matching points on the graph, demonstrating a close value of throughput. The improvement over BEB varies from 5.19% for 5 stations (i.e., with an increase of 17.64% for 15 stations and 28.70% for 30 stations) to 48.5% for 50 stations. As can be seen, DDPG has a slightly better performance than DQN (with either one of the observation metrics), which can be explained due to its capability to choose any real CW value within the $[0, 6]$ range. In the static scenario, the performance achieved by the solution employing either one of the two observation metrics shows a minimal difference. When compared, there is no significant variance in their behavior. Moreover, the experiment considers a level of significance (α) of 0.05 for a 95% confidence interval.

The figures 4.3 through 4.6 present the mean CW value and its variance for 15 simulation episodes when DQN or DDPG is used with either one of the two observation metrics considered. Each figure considers the experiment from 5 to 50 stations in the static scenario. The mean CW serves as an indicator of how well the training agents can adjust and select the CW values to a more stable one. It shows that as the agent learns, it exploits more of the acquired/learned knowledge than explores the environment with random actions. Therefore, as the DRL agent learns, the number of random actions decreases, decreasing the variance of the CW values.

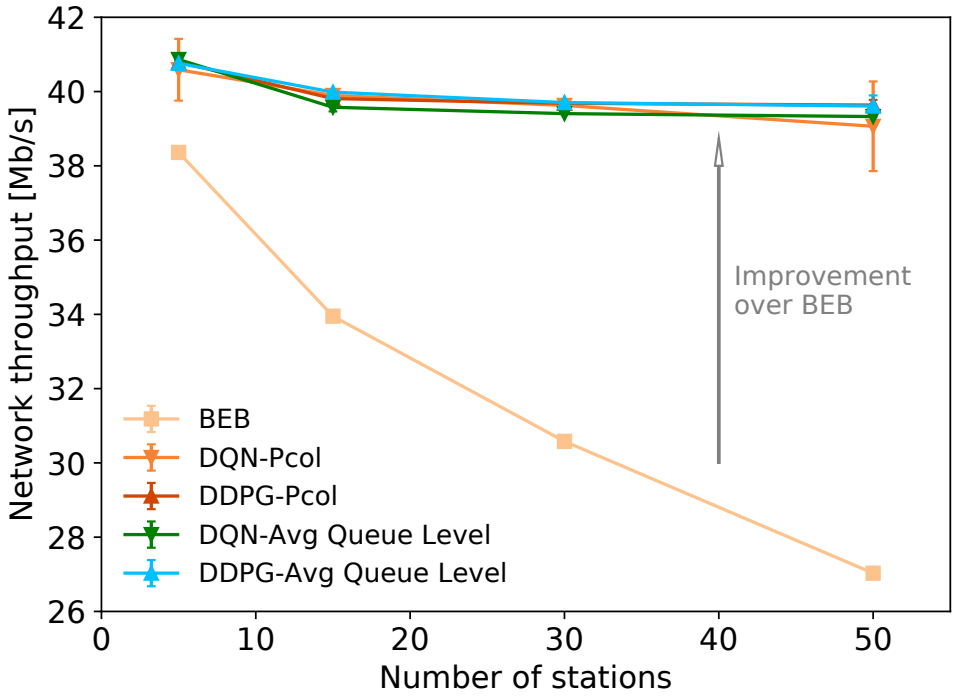

Figure 4.2: *Comparison of the network throughput for the static centralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol and Avg queue level as observations.*

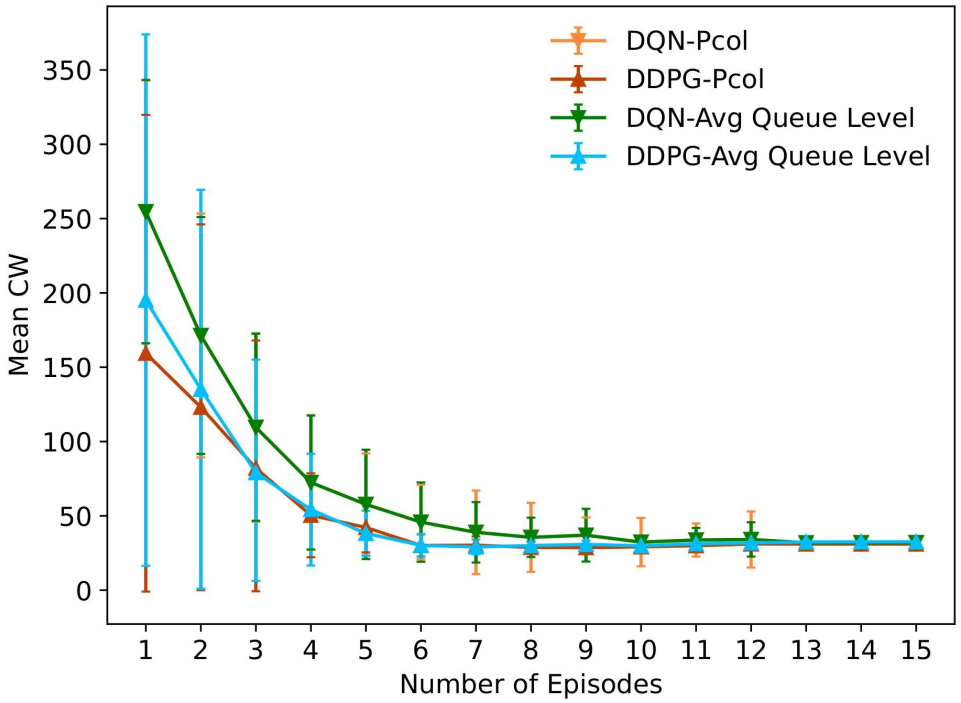

Figure 4.3: *Mean CW value for 5 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

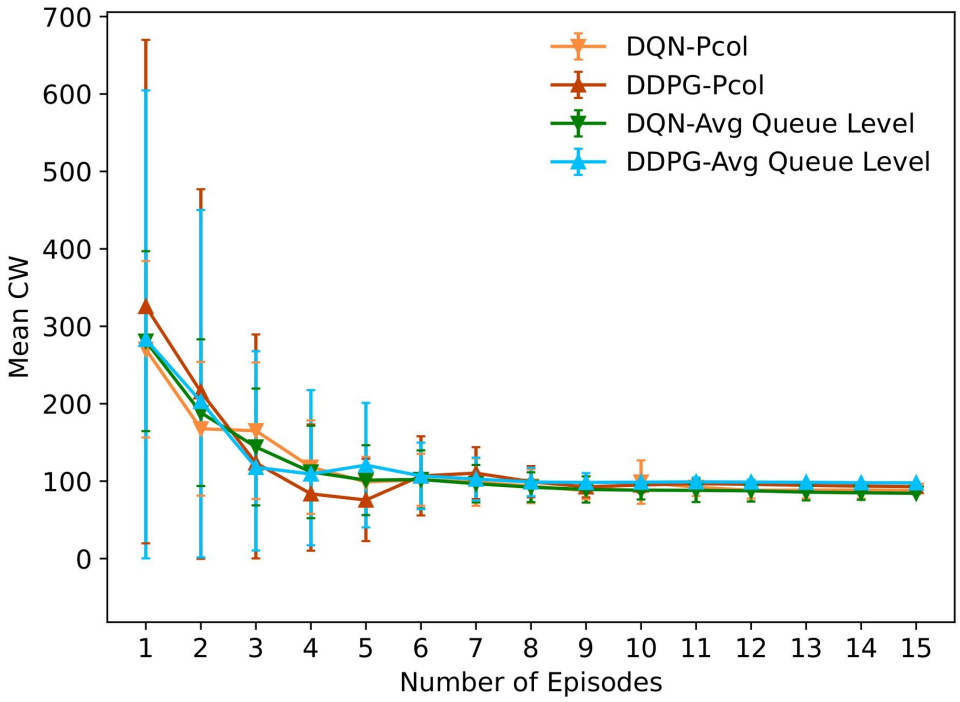

Figure 4.4: *Mean CW value for 15 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

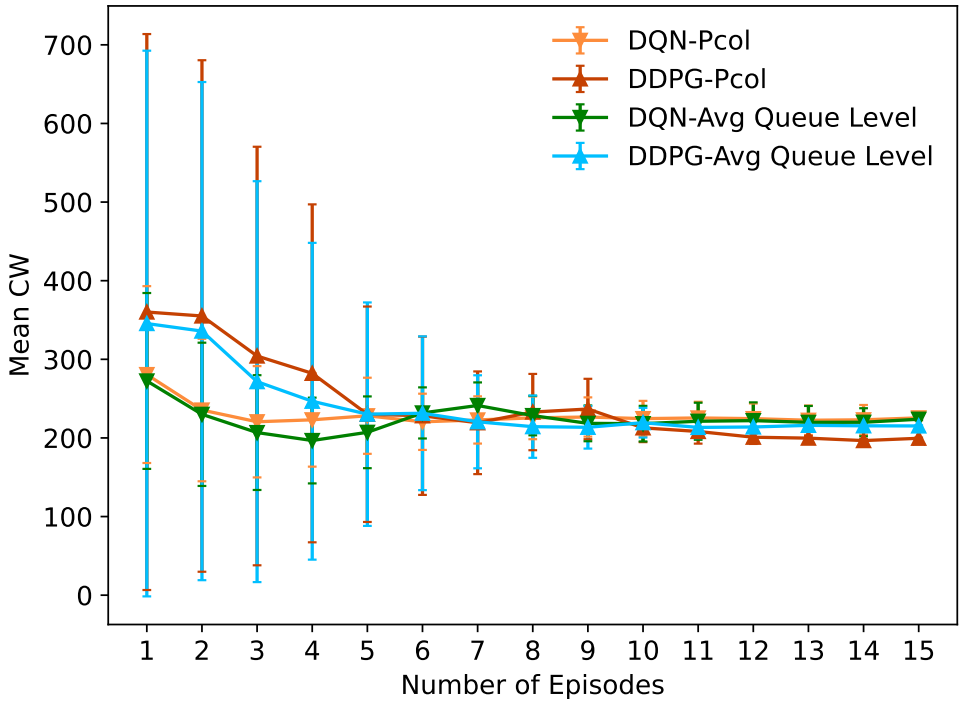

Figure 4.5: *Mean CW value for 30 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

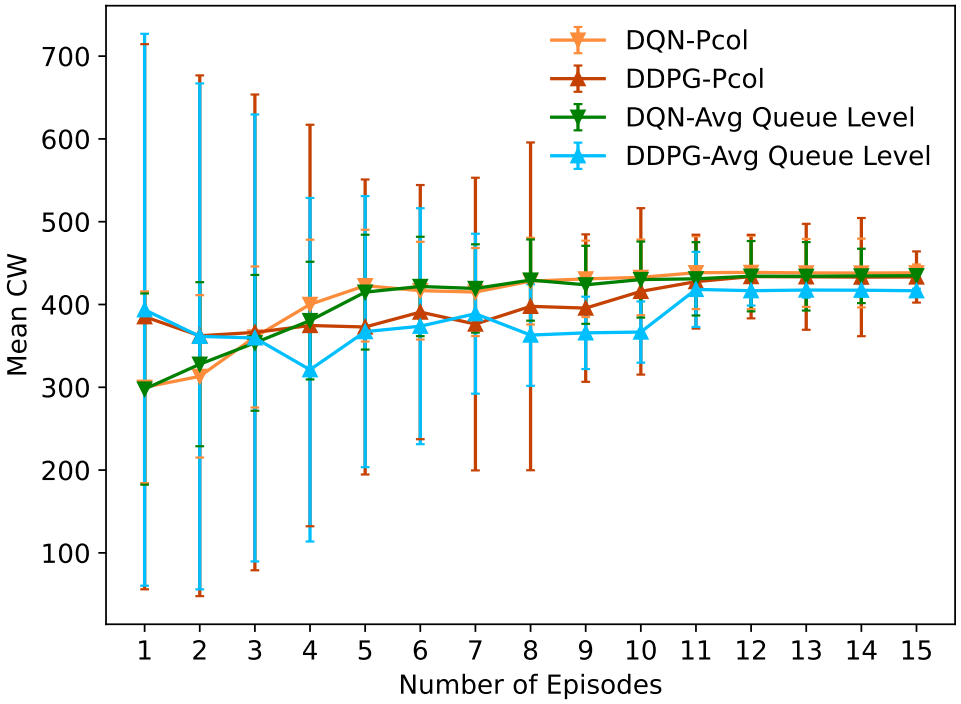

Figure 4.6: *Mean CW value for 50 stations in a static centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

It is pretty clear that the 14 episodes selected for the learning stage are enough for the proposed solution to converge to an optimal and practically constant mean CW value for both DRL algorithms, regardless of the observation metric considered. We can see that the mean value of all considered options (DQN/DDPG and average transmission queue level/Pcol) stabilizes around the same value after the 10th episode. It is also possible to see that the CW variance of all options decreases along the learning stage, meaning that initially, the algorithm explores the environment more (i.e., it takes uncharted actions). 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. In the final episodes, where the algorithm exploits more than explores the environment, DDPG presents a smaller variance than DQN, meaning it is more assertive in choosing the ideal CW value.

Dynamic scenario results

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.

Figures 4.7 to 4.10 present the mean CW and variance values for the dynamic scenario through 15 simulation episodes when DQN or DDPG is used with any of the two observation metrics considered (Pcol and average queue level) considering 5, 15, 30, and 50 stations. In both observation metrics, the results demonstrate that in the first episodes, there is a higher variance on CW, but as the training progresses, there is less variance. This means that the number of random actions decreases over time and the agent converges to a result and learning correctly. Furthermore, regarding the mean CW, there is not much difference amongst the experiments with 5, 15, 30, and 50 stations using Pcol and average queue level using DQN and DDPG in terms of graphical behavior as it stabilizes converging to a specific value.

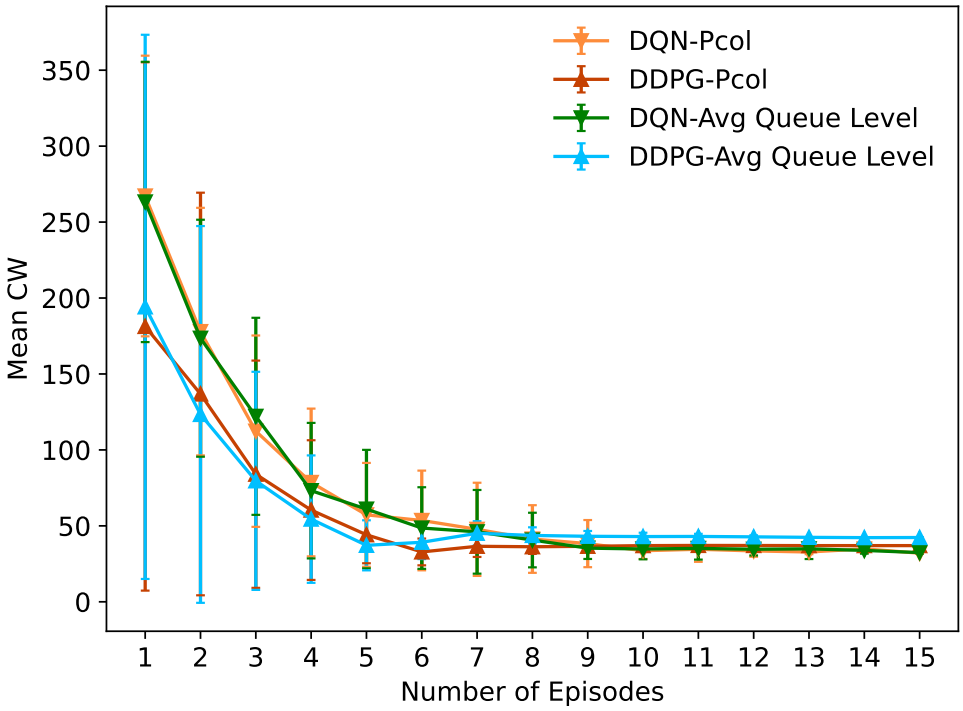

Figure 4.7: *Mean CW value for 5 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

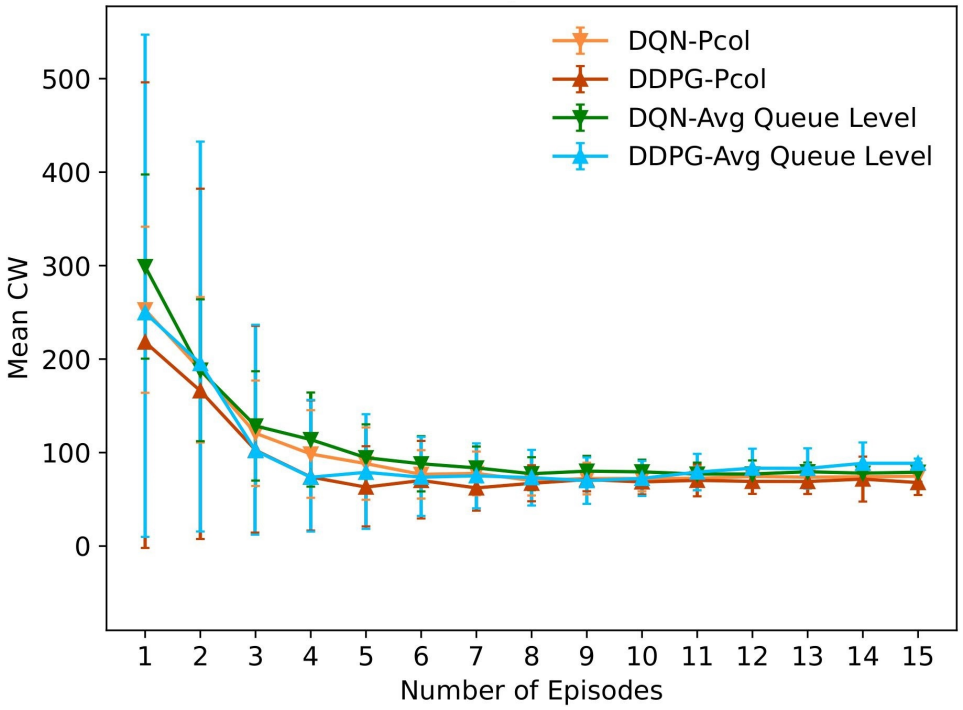

Figure 4.8: *Mean CW value for 15 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

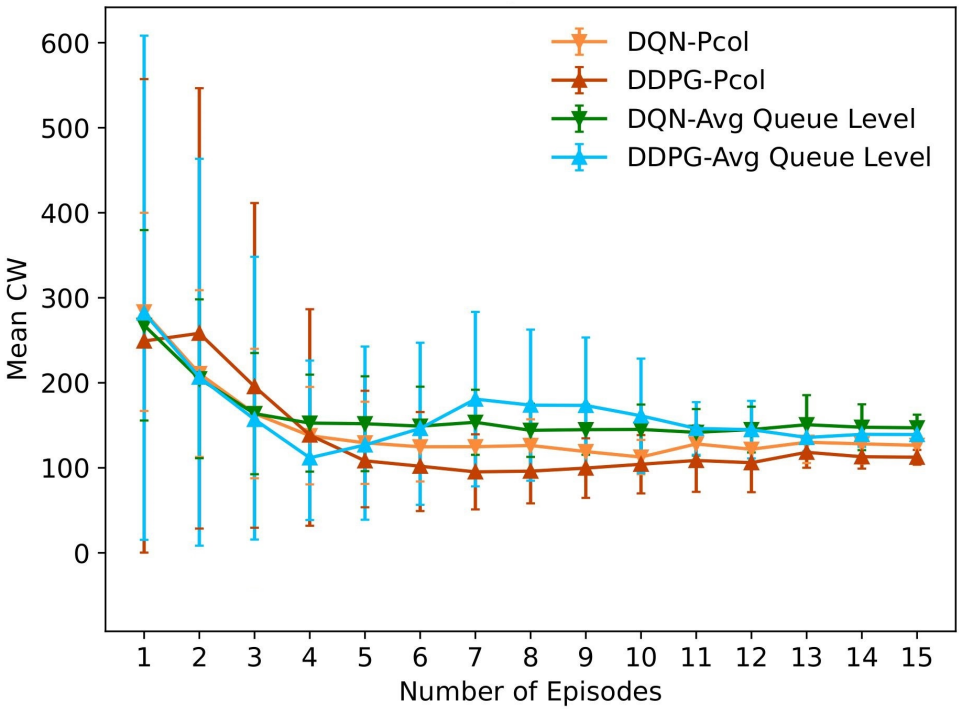

Figure 4.9: *Mean CW value for 30 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

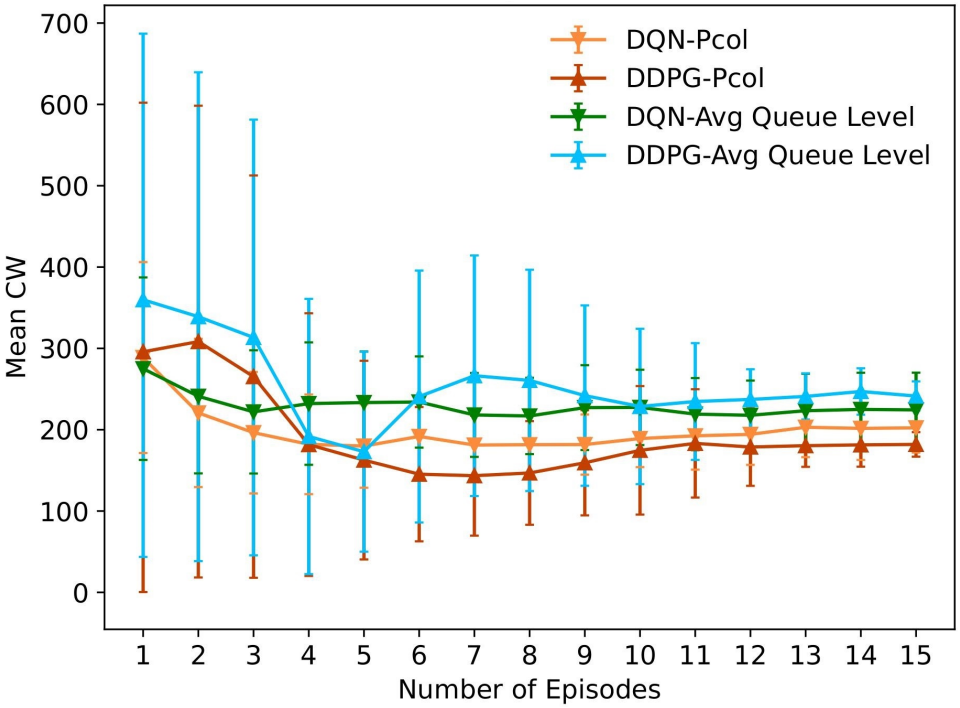

Figure 4.10: *Mean CW value for 50 stations in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

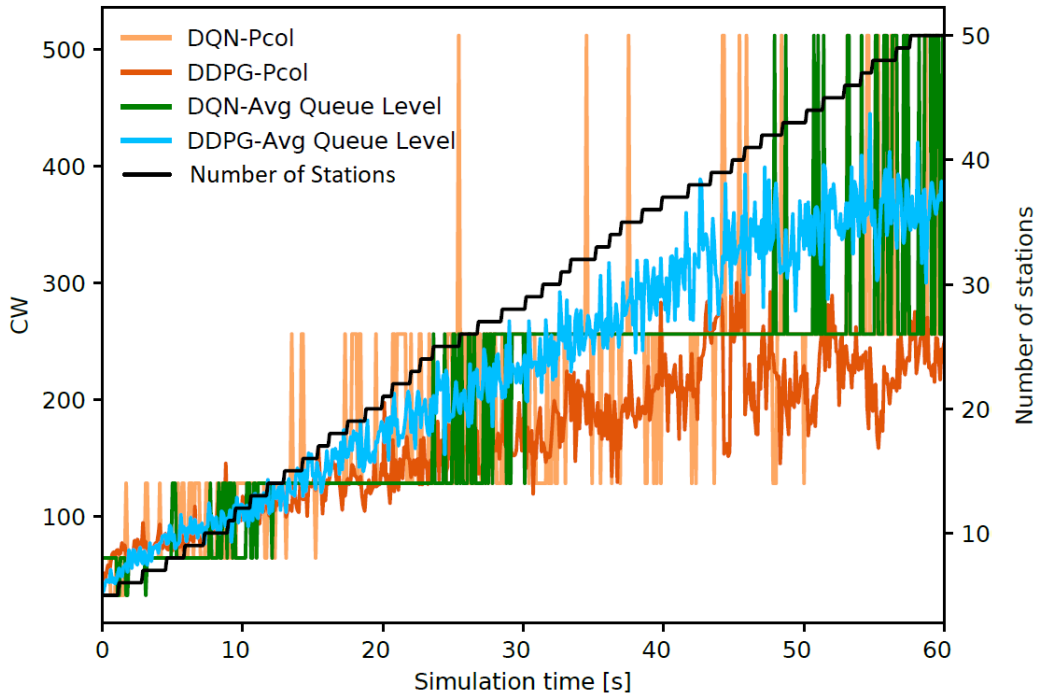

Figure 4.11: *Selected CW value with the number of stations increasing from 5 to 50 in a dynamic centralized scenario using DQN, and DDPG with Pcol and Avg queue level as observations.*

Figure 4.11 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 independently of the adopted observation metric, which positively reflects on the attained throughput. When comparing DQN for both observation metrics, it is possible to see a similarity in selecting the adequate CW value. Furthermore, DDPG with the averaged normalized transmission queues level as observation presents higher CW values than when the probability of collision is used. This behavior, which is supported by the results in Figure 4.12, can be partially explained by the fact that the normalized transmission queues level of all associated stations offers more information than just the collision probability. This metric takes into account the congestion level of the network and subsequently helps the DRL-based solution select a CW value that allows the network to achieve higher throughput. This insight suggests that the use of this metric in the RL solution for collision avoidance could result in improved network performance.

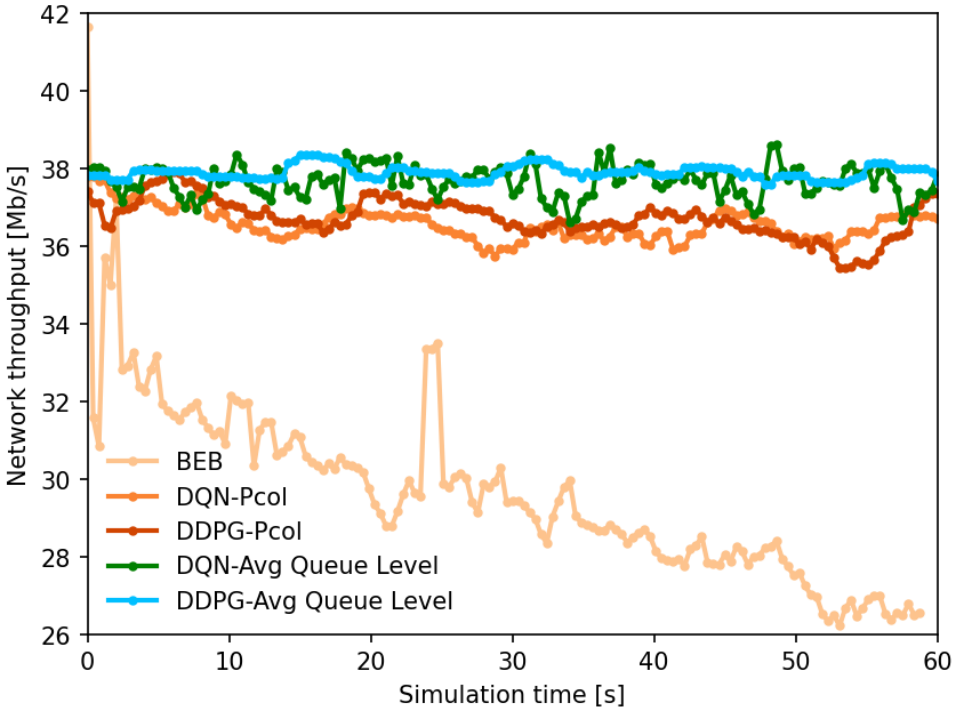

Figure 4.12: *Comparison of the instantaneous network throughput in a dynamic centralized scenario using BEB, DQN, and DDPG with Pcol and Avg queue level as observations.*

Throughput and Collision comparison with an increasing number of stations

Figure 4.12 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 26 Mbps 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 number of stations progressively increases. Both DRL algorithms present an approximately constant throughput. Comparing the two observation metrics reveals a difference in the network throughput. Using the collision probability as the observation, the resulting throughput is approximately 37.5 Mbps, presenting an increase of 43.68% over BEB, whereas using the averaged normalized transmission queues level yields a throughput of up to 37.98 Mbps with an increase of 45.52% over BEB. This difference suggests that the averaged normalized transmission queues level is a superior observation metric for dynamic scenarios. However, it is essential to note that both observation metrics achieve significantly better throughput values than the standard BEB algorithm. This finding underscores the potential value of utilizing RL solutions for collision avoidance in Wi-Fi networks.

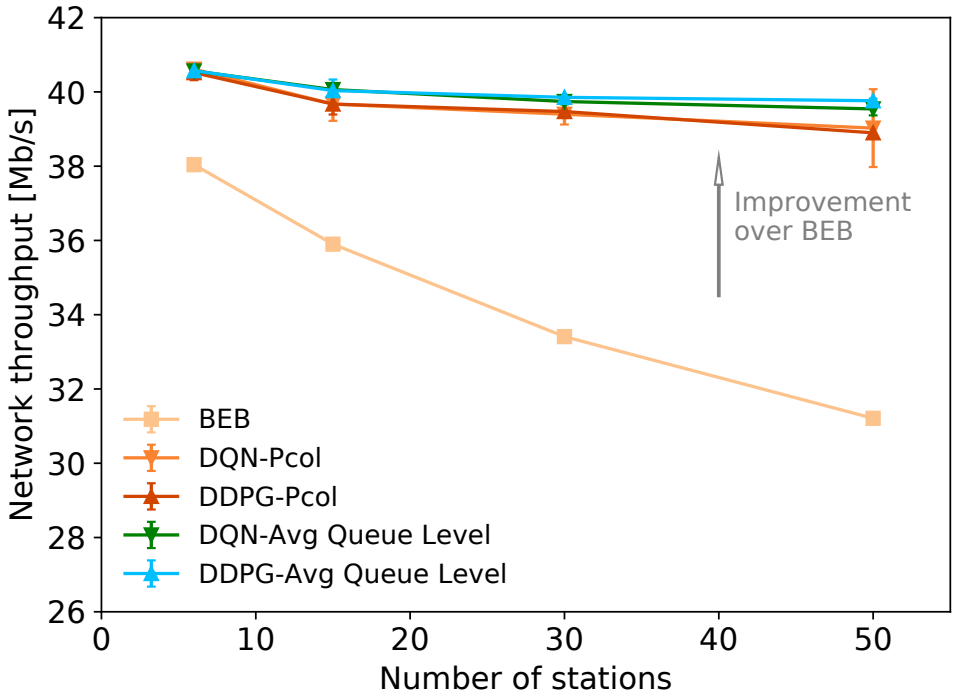

Figure 4.13: *Comparison of the network throughput for the dynamic centralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol and Avg queue level as observations.*

Figure 4.13 confirms that both DRL algorithms significantly enhance the throughput of the network in comparison to the BEB algorithm in dynamic scenarios. The degree of improvement varies from 6.84% (using either observation metric) for 5 stations, 10.83% for 15 stations, 19.28% for 30 stations, and 25.56% for the collision probability metric when 50 stations are in use. Notably, when the averaged normalized transmission queues level is employed with 15 stations, there is an increase of 11.84%, 20.28% for 30 stations, and for 50 stations, the throughput is enhanced by as much as 28.43%, thereby indicating the superior performance of this observation metric over the collision probability in dynamic scenarios with 50 stations. The experiment result considers a level of significance (α) of 0.05 for a 95% confidence interval. These findings demonstrate once more the potential value of using DRL algorithms to address collision avoidance challenges in dense Wi-Fi networks.

4.0.3 Decentralized Solution Validation

To validate the proposed solution, several simulation sanity tests were performed with the decentralized topology to ensure the consistency of the code and whether the decisionsof the agent made sense following a logical structure. The validation test comprised five stations considering the static and dynamic scenarios with three different types of actions, namely the RL action of the agent (i.e., DQN and DDPG), random action, and constant action. With the action of the agent, the agents are able to learn and obtain rewards that maximize the overall cumulative reward and improve the instantaneous network throughput. The Random action is a random integer array from the Numpy library in Python that follows the interval of [0,6]. The constant action was set to a fixed constant integer number.

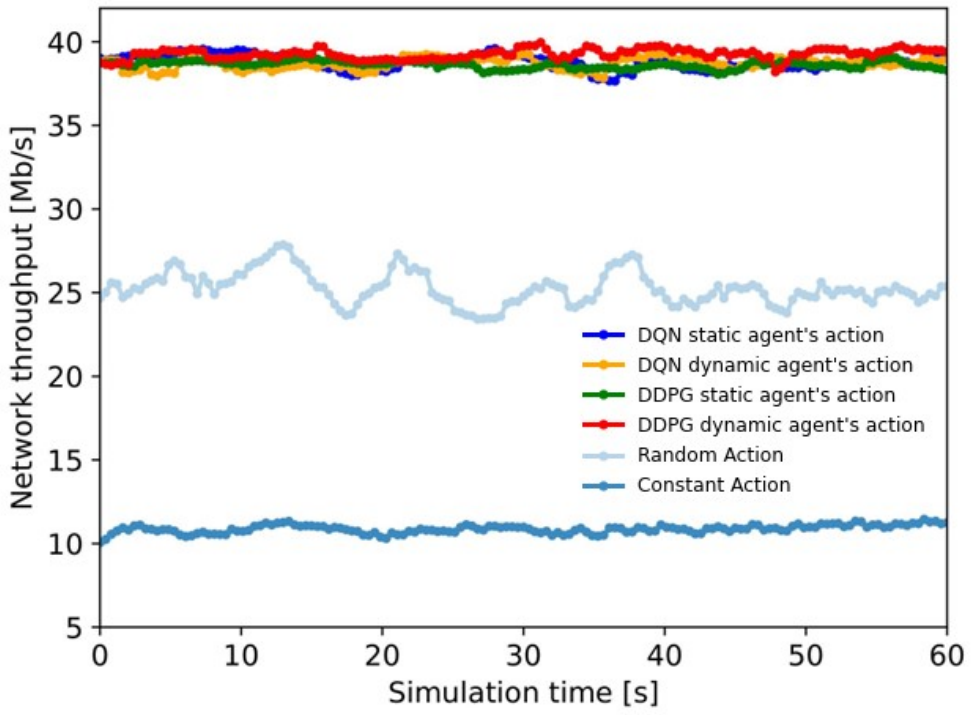

Figure 4.14: *Validation test comparing the instantaneous network throughput for the decentralized static and dynamic scenarios using different action types, including DQN, DDPG, random, and constant.*

Figure 4.14 presents the obtained instantaneous network throughput for each action type considering the static and dynamic scenarios for the RL agents during 15 episodes, it can be observed that the RL actions present better throughput values than random or constant actions indicating that the RL agent can learn to make optimal decisions to improve the performance of the network in a multi-agent mode reaching up to about

39.50 Mbps of throughput with 5 stations in either scenario. Since the type of action influences the generated reward figures 4.15 and 4.16 present the obtained cumulative reward, and it is possible to see that when the RL algorithms are applied, the cumulative reward greatly increases than random or constant actions. The obtained results from the sanity tests have motivated us to the final implementation of the proposed approach.

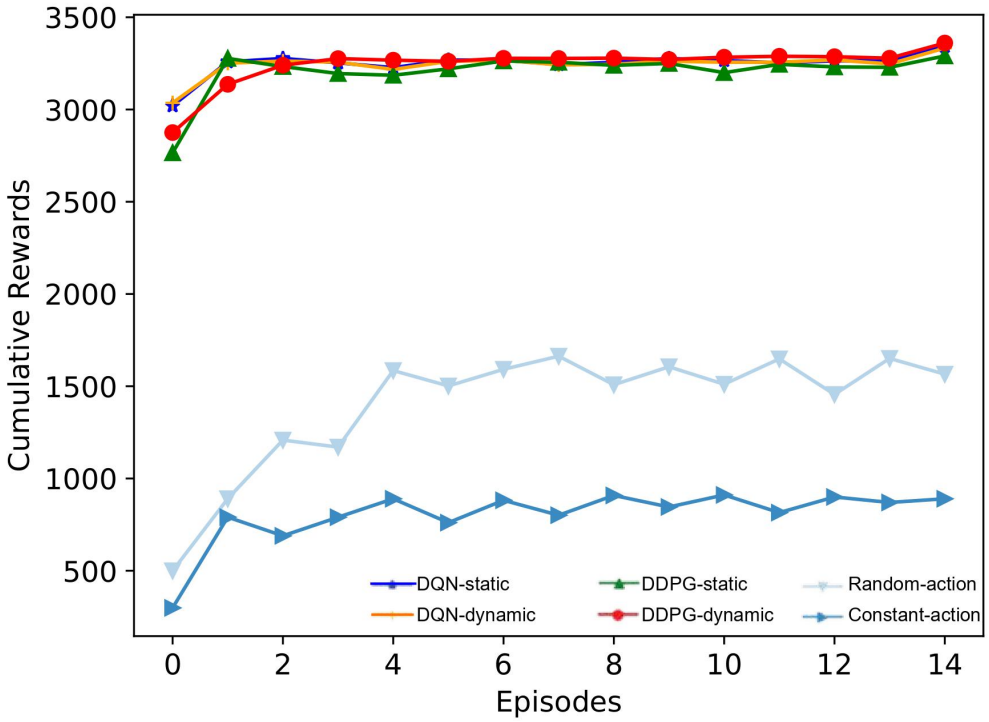

Figure 4.15: *Validation test comparing the cumulative reward for the decentralized static and dynamic scenarios using different action types, including DQN, DDPG, random, and constant.*

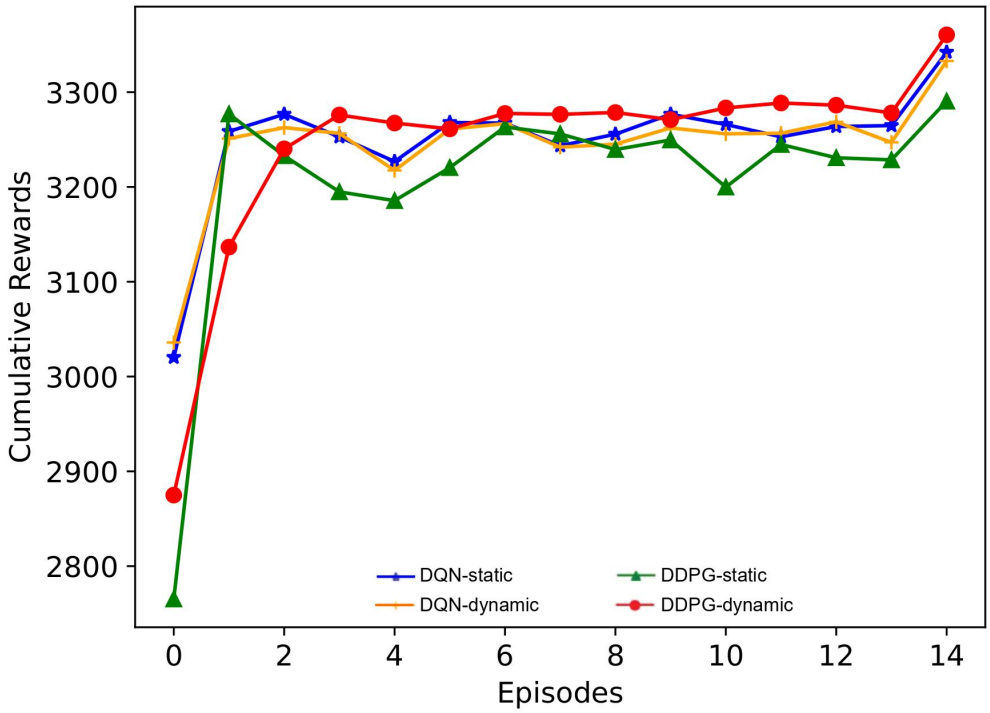

Figure 4.16: *Validation test comparing the cumulative reward for the decentralized static and dynamic scenarios using DQN and DDPG.*

4.0.4 Centralized vs Decentralized Results

Static scenario results

Figure 4.17 confirms that both DRL algorithms significantly enhance the throughput of the network in comparison to the BEB algorithm in the centralized and decentralized static scenarios. The degree of improvement in comparison to BEB with the probability of collision as the same observation metric was similar, except for 30 and 50 stations. For the centralized and decentralized approach with 5 stations, the improvement over BEB is 5.19%, for 15 stations, the improvement is 17.64%. For 30 stations, the centralized scenario has a 33% improvement, and the decentralized has a 16.67% improvement, for 50 stations, the centralized scenarios have a 47.41% improvement, and the decentralized has a 27.78% improvement. The results demonstrate that DDPG is slightly better than DQN in the centralized and decentralized solution, especially for 50 stations. Furthermore, the centralized solution has a better performance than the decentralized one. Moreover, the experiment considers a level of significance (α) of 0.05 for a 95% confidence interval.

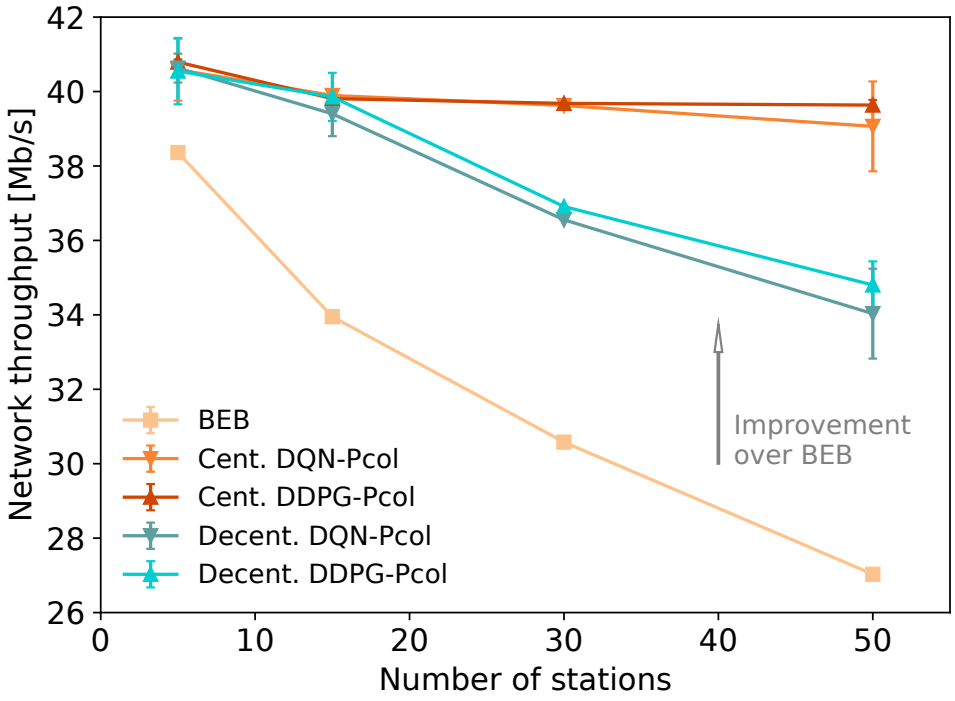

Figure 4.17: *Comparison of the network throughput for the static centralized and decentralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol as the observation.*

Figures 4.18 to 4.21 present the mean CW and variance values for the static centralized and decentralized solution through 15 simulation episodes when DQN and DDPG are used considering 5, 15, 30, and 50 stations. The results demonstrate that in the initial episodes, there is a higher variance on CW, but as the training progresses, the variance reduces. This happens because the number of random actions decreases over time, and the agent is converging to a result and learning correctly. Furthermore, regarding the mean CW, for 5 stations, the decentralized DDPG has lower CWs than the other approaches. For 15 to 50 stations, the centralized approaches converge to lower CWs when compared to the decentralized approaches and the centralized DDPG had the overall lowest mean CW. As shown in Figure 4.21 the decentralized result for DQN and DDPG has visibly more variation than the others (4.18 to 4.20) due to the training process which takes much longer for having 50 stations sharing the same environment, this leads to an unstable environment. In the decentralized scenario, each station has its individual CW value being optimized, so each station ends up competing with one another to maximize its reward by acting more selfishly which is justified by game theory.

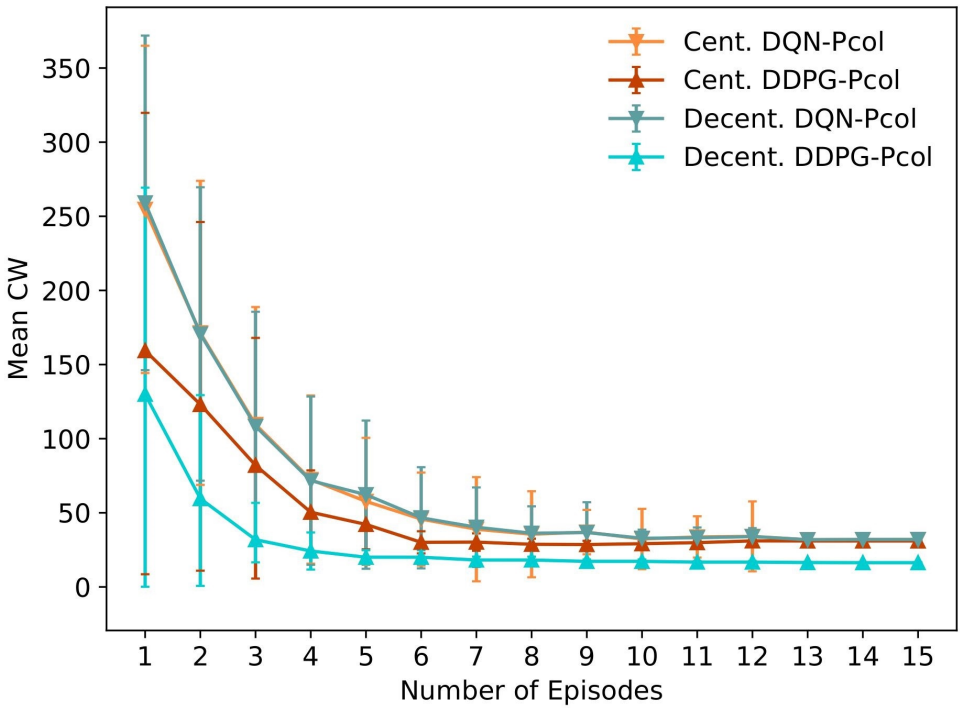

Figure 4.18: *Mean CW value for 5 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

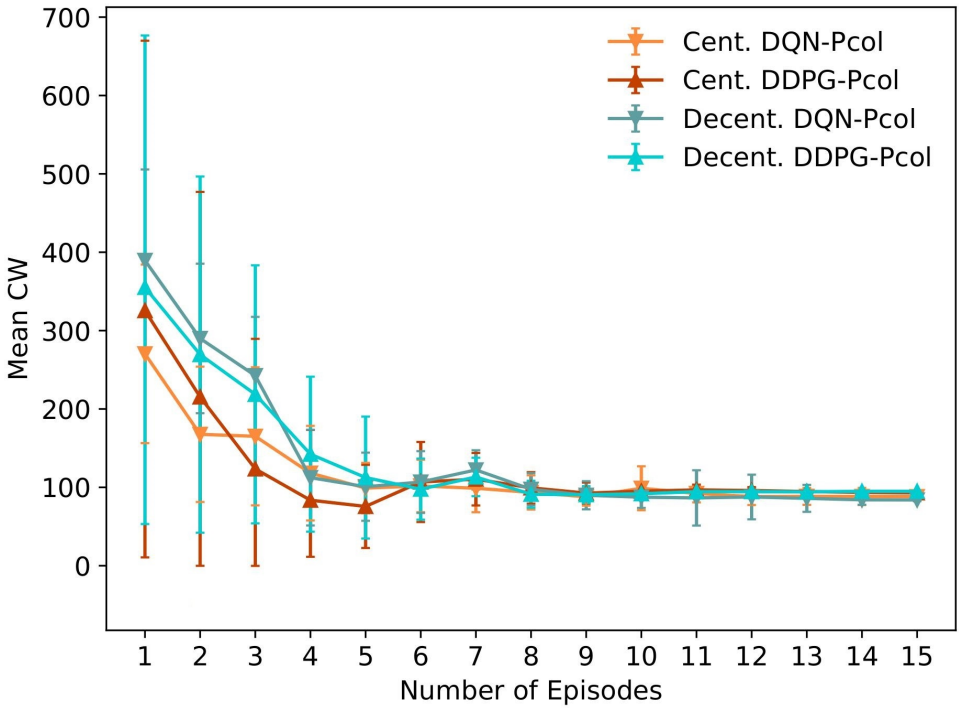

Figure 4.19: *Mean CW value for 15 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

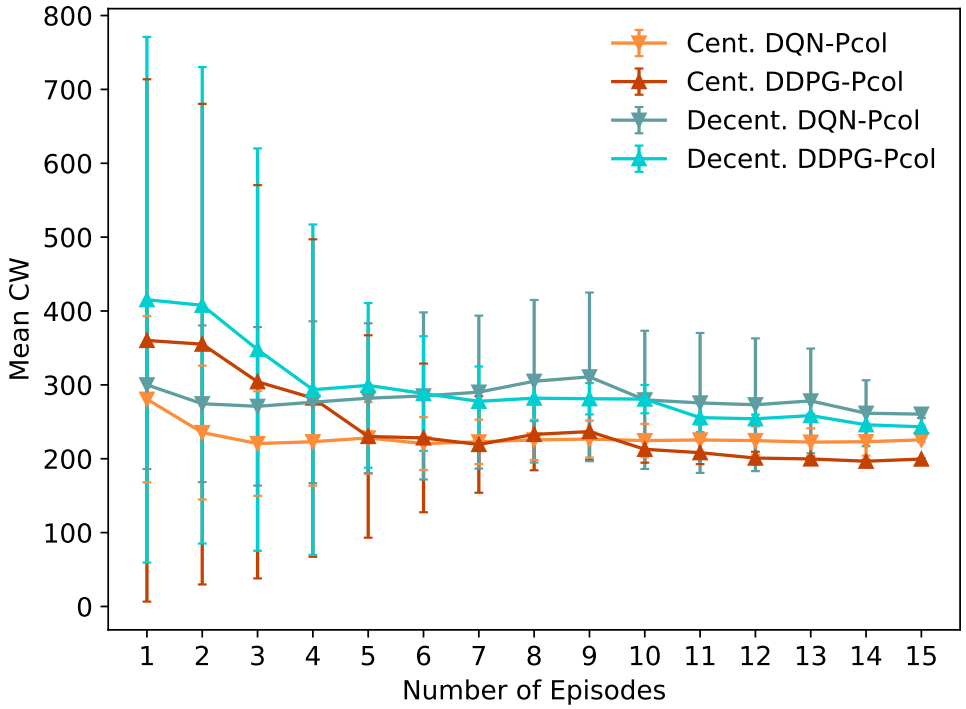

Figure 4.20: *Mean CW value for 30 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

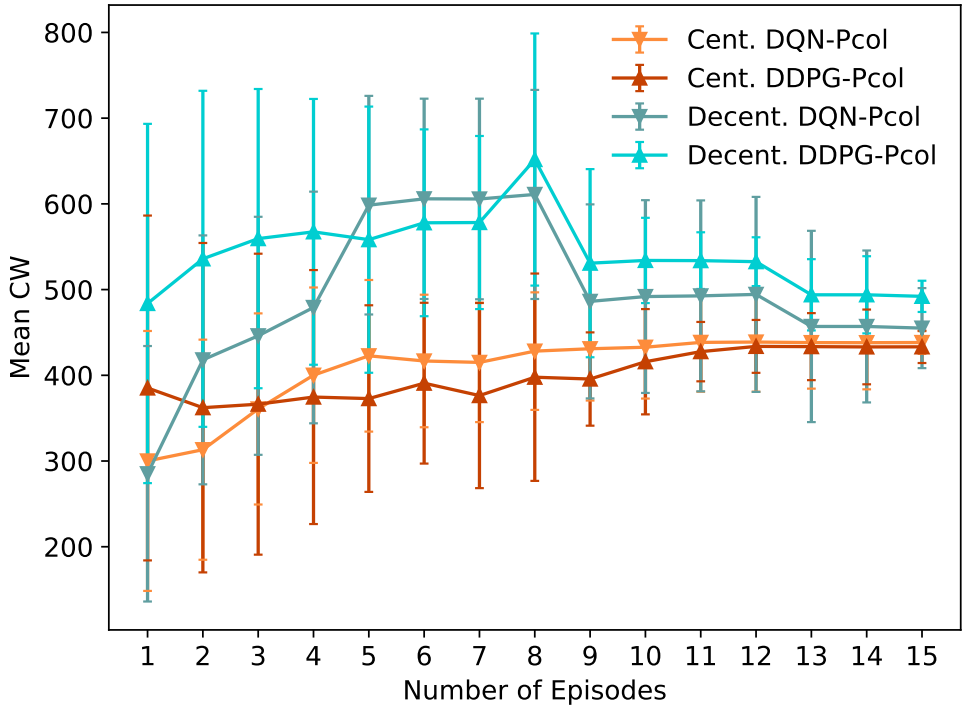

Figure 4.21: *Mean CW value for 50 stations in a static centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

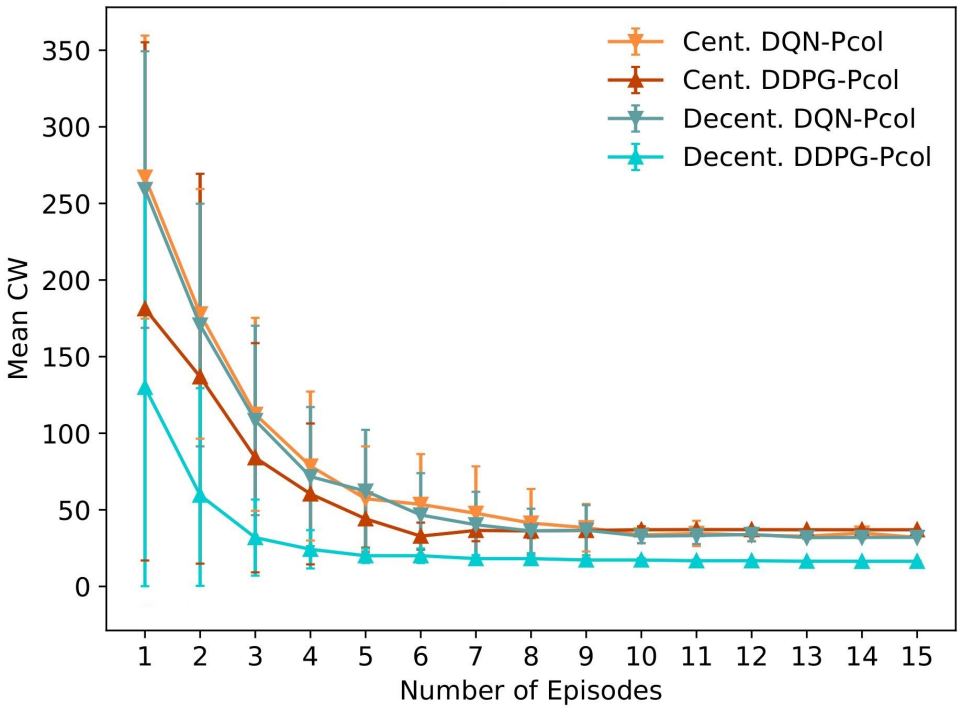

Figure 4.22: *Mean CW value for 5 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

Dynamic scenario results

Figures 4.22 to 4.25 present the mean CW and variance values for the centralized and decentralized dynamic scenario through 15 simulation episodes when DQN and DDPG are used considering 5, 15, 30, and 50 stations. The results demonstrate that in the initial episodes, there is a higher variance on CW, 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. Furthermore, regarding the mean CW, for 5 stations, the decentralized DDPG has lower CW values than the centralized one. For 15 to 50 stations, the centralized approach converged to lower CWs when compared to the decentralized solution and the centralized DDPG had the overall lowest mean CW.

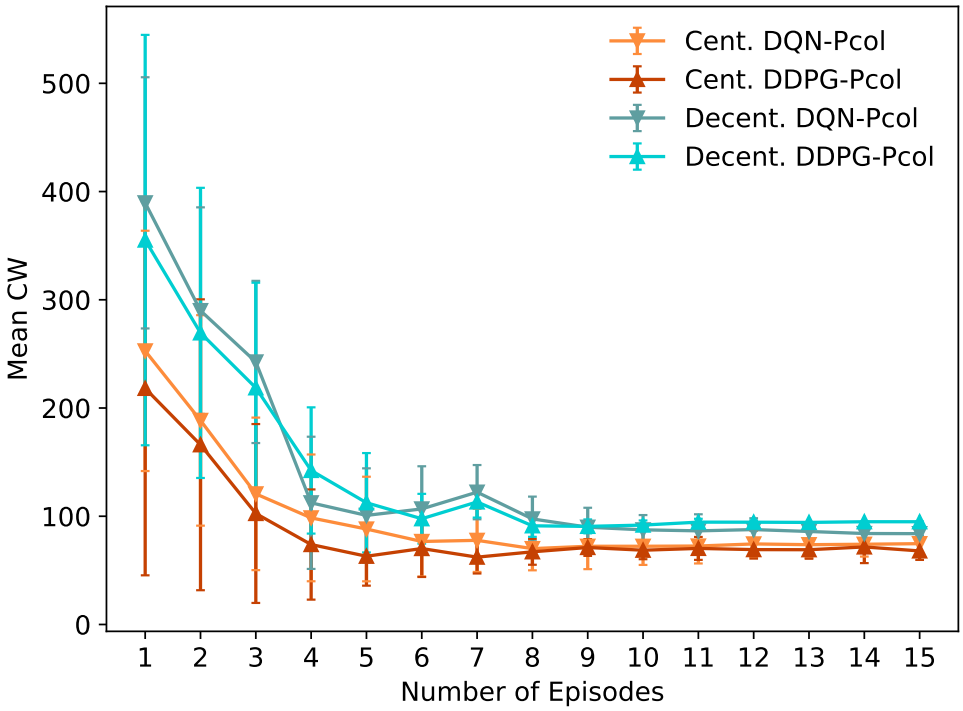

Figure 4.23: *Mean CW value for 15 in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

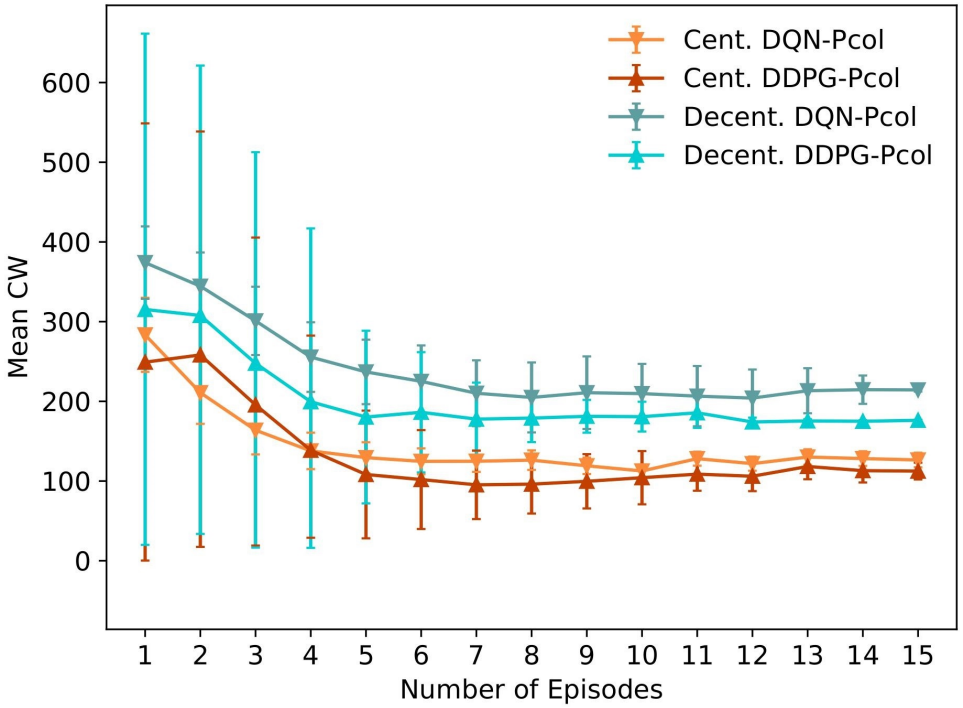

Figure 4.24: *Mean CW value for 30 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

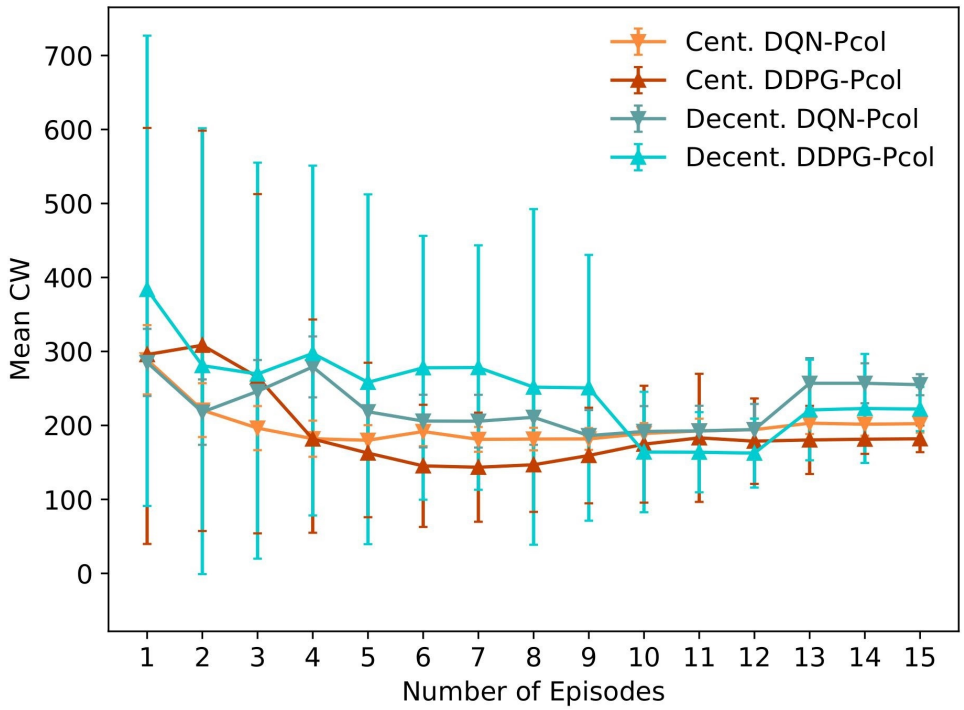

Figure 4.25: *Mean CW value for 50 stations in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

Throughput and Collision comparison with an increasing number of stations

Figure 4.26 illustrates the mean CW values calculated for both dynamic centralized and decentralized scenarios over 50 stations with the number of stations as a function of simulation time incrementally rising from 5 to 50 at every 1.2 seconds. As the number of stations increases, the CW values are adjusted accordingly. It is evident that DQN varies between discrete neighboring CW values, while DDPG consistently raises the CW value, resulting in a lower CW value for 50 stations and subsequently causing an improvement in throughput. Comparing the two DQN (i.e., centralized and decentralized) agents, there is a notable resemblance in how they choose the appropriate CW value.

Figure 4.27 compares the instantaneous network throughput in the dynamic centralized and decentralized scenario, where the number of stations progressively increases 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 26 Mbps 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 number of stations progressively increases. Both DRL algorithms present an approximately constant throughput. Using the collision probability as the observation,

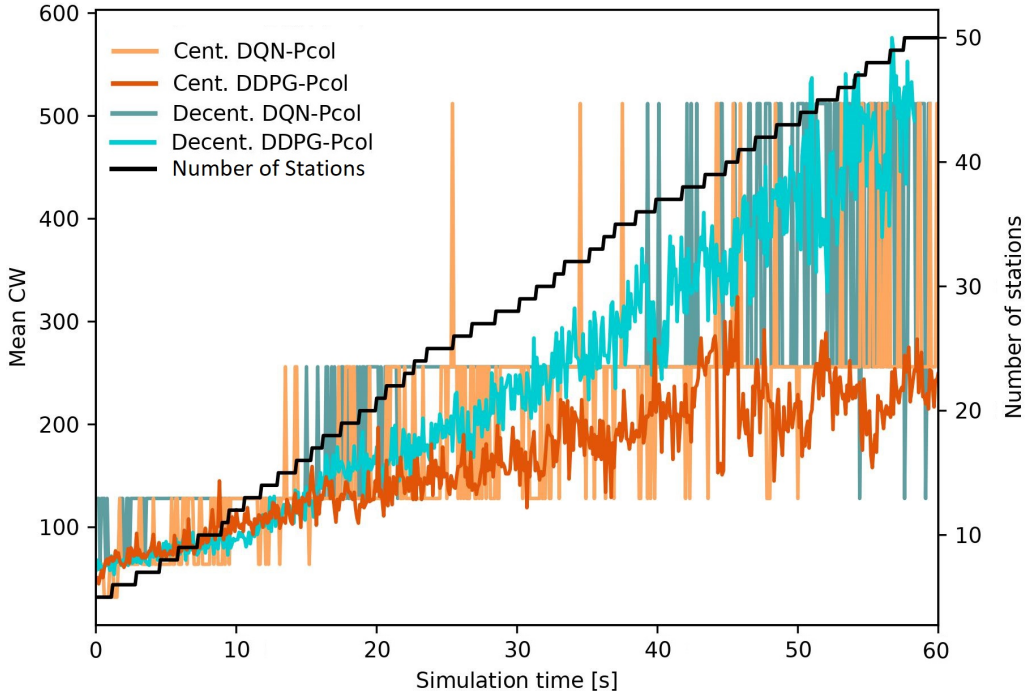

Figure 4.26: *Mean CW value with the number of stations increasing from 5 to 50 in a dynamic centralized and decentralized scenario using DQN and DDPG with Pcol as observation.*

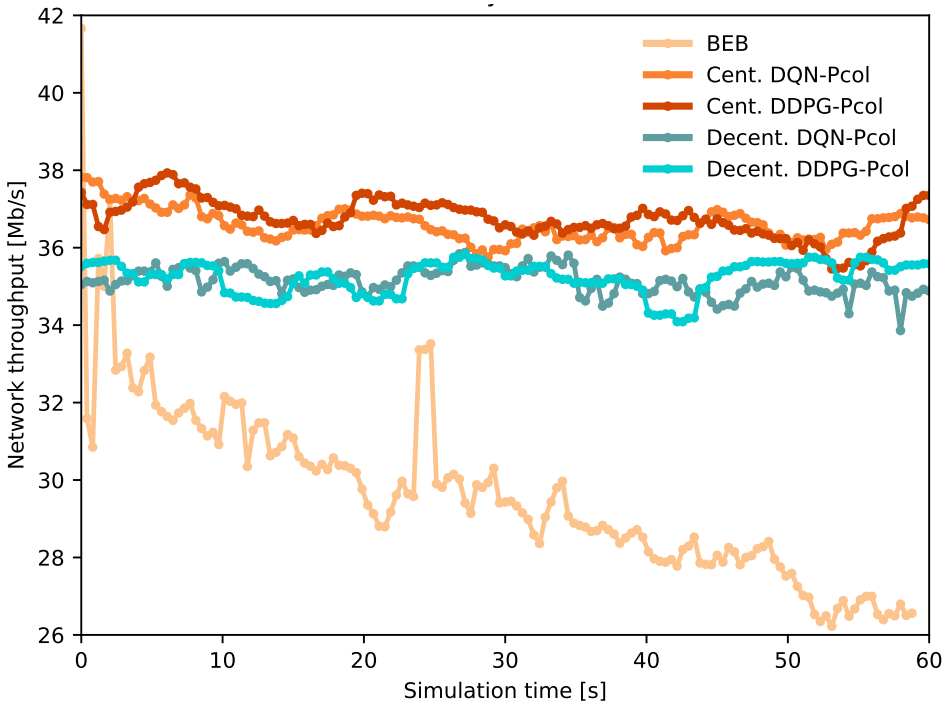

Figure 4.27: *Comparison of the instantaneous network throughput in a dynamic centralized and decentralized scenario using BEB, DQN, and DDPG with Pcol as observation.*

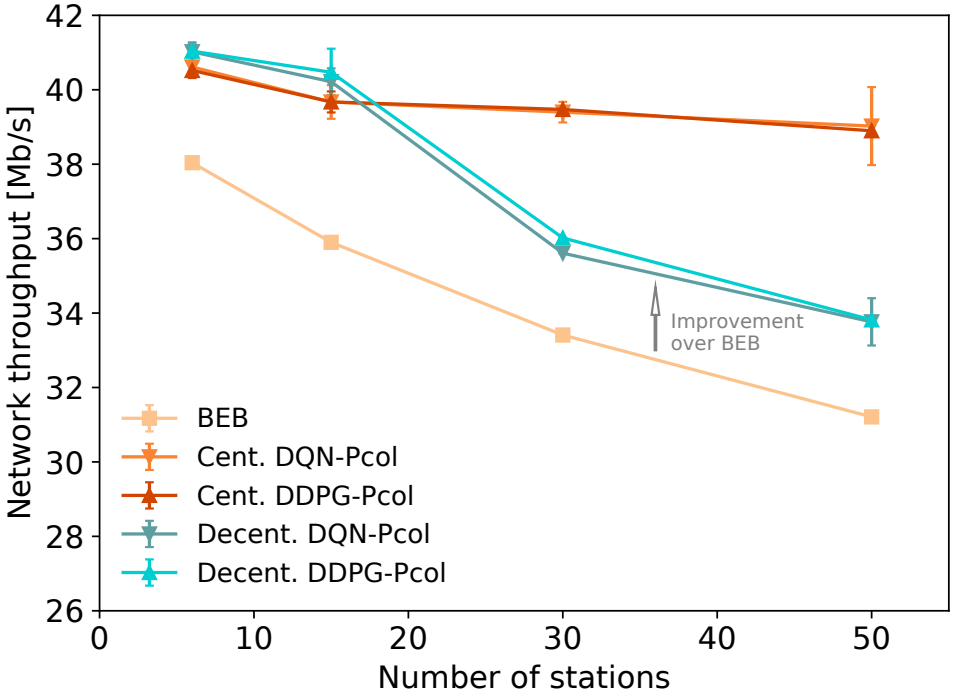

Figure 4.28: *Comparison of the network throughput in a dynamic centralized and decentralized scenario as the number of stations increases from 5 to 50 using BEB, DQN, and DDPG with Pcol as observation.*

the resulting throughput of the centralized approach is approximately 37.5 Mbps, presenting an increase of 43.68% over BEB, whereas using the same collision probability as the observation in a decentralized mode yields a throughput of 35.8 Mbps with an increase of 37.16% over BEB. Despite that, the decentralized solution presents a lower throughput value than the centralized one, it is important to mention that both solution approaches can achieve significantly better throughput values than the standard BEB algorithm. This finding underscores the potential value of utilizing RL solutions for collision avoidance in Wi-Fi networks.

Figure 4.28 confirms that both DRL algorithms significantly enhance the throughput of the network in comparison to the BEB algorithm in the centralized and decentralized dynamic scenario. The degree of improvement in comparison to BEB with the collision probability as the observation metric was similar, except for 30 and 50 stations. For the centralized case for 5 stations, the improvement is 6.84%, for 15 stations, the improvement is 10.83%, for 30 stations is a 19.28% improvement, and for 50 stations, a 25.56% improvement over BEB. In the decentralized mode, the improvement for 5 stations is 7.89%, 15 stations have an 11.94% improvement, 30 stations have a 9.27%, and for 50 stations, an 8.43% improvement over BEB. The results demonstrate that DDPG is slightly better than DQN in the centralized and decentralized solutions, especially for 30 and 50 stations. Furthermore, centralized solutions have a better

performance than decentralized ones. Moreover, the experiment considers a level of significance (α) of 0.05 for a 95% confidence interval.

In the decentralized solution, the RL agents located within the stations compete with each other. As there is no information sharing between them, this impacts the resulting throughput value, as it is not being balanced between the stations. Consequently, some stations end up with a higher throughput than others. The lack of cooperation between RL agents generates competitive behavior between stations, which is justified by game theory [107].

Game Theory is an analytical modeling tool used in many application areas for decision-making problems [107] [108], including wireless networks for resource allocation, access control, collision avoidance, rate adaptation, and so forth. Its objective is to identify an optimal strategy for each participant (i.e., player) in a given scenario or situation. In game theory, in the non-cooperative case, participants do not have access to information about each other. Without cooperation, no information is shared, which leads participants to be greedy and selfish as they try to get the best reward individually. In other words, each participant seeks to gain a competitive advantage over the others.

Chapter 5

Conclusions and Future Work

The thesis addresses the inefficiencies in the collision avoidance mechanism of the IEEE 802.11 standard, specifically the drawbacks of the binary exponential backoff (BEB) algorithm in the medium access control (MAC) layer. The conventional approach of increasing backoff intervals upon collision detection results in significant radio spectrum utilization degradation, especially in dense network scenarios, leading to a substantial decline in overall network performance.

To overcome these challenges, the thesis proposes the application of reinforcement learning (RL) algorithms, specifically Deep Q Learning (DQN) and Deep Deterministic Policy Gradient (DDPG), as effective solutions to optimize the contention window (CW) value. The proposed approach evaluates two observation metrics, namely the average of the normalized level of the transmission queue of associated stations and the probability of collisions (Pcol), with the overall network throughput serving as the reward. Simulations conducted using the NS-3 network simulator and NS3-gym toolkit demonstrate that both DQN and DDPG consistently outperform BEB.

In addition to demonstrating the superiority of DQN and DDPG over the BEB algorithm, the thesis takes into consideration various network scales (considering 5, 15,30, and 50 stations) and two operational scenarios (static and dynamic). The evaluation in both static and dynamic scenarios takes into consideration the same number of stations for both scenarios. Remarkably, the proposed RL-based approach demonstrated enhanced performance regardless of the number of stations and operational scenarios, indicating its adaptability to diverse network scales. Furthermore, the assessment encompasses both single-agent (centralized) and multi-agent (decentralized) approaches, highlighting the flexibility of DQN and DDPG in optimizing contention window values under different network architectures. This comprehensive analysis demonstrates the robustness of the reinforcement learning algorithms in addressing the challenges posed

by the collision avoidance mechanism of the IEEE 802.11 standard across a spectrum of realistic deployment scenarios. Therefore, the thesis successfully contributed towards the advancement of the state of the art in network optimization using machine learning.

5.0.1 Chapter Summary

The thesis presents an updated study on optimizing the contention window using machine learning, and improvements were made to the state of the art on the topic. The first chapter introduced the motivation and delimited the research topic. The main and partial goals were then described. The chapter ended by presenting the main contributions and publications.

Chapter 2 provided the background for the main research topic. It starts with an overview of previous studies that attempted to optimize the contention window using machine learning. Then, the main difference between this study and the previous work available in the literature is explained. Next, an overview regarding machine learning, as well as the differences between supervised learning, unsupervised learning, reinforcement learning, and multi-agent reinforcement learning was presented, and details were provided as to why reinforcement learning was the chosen paradigm for the task at hand. The chapter ends by detailing Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) that were used later in the experiments.

Chapter 3 delved into the proposed methodology, detailing the experiments to ensure their reproducibility. The chapter unveiled the algorithms for both the centralized and decentralized approaches,

Chapter 4 unveils the configuration parameters chosen for the experiment. This in-depth exploration provides a comprehensive understanding of the experimental setup, enabling researchers to replicate the experiments and validate the findings. To enhance the clarity and reproducibility of the experiments, the chapter includes a high level of detail for the parameters utilized in each algorithm, fostering a foundation for further research and validation. This chapter also presents the results obtained from the extensive experiments, starting with the centralized solution and culminating in the decentralized solution. For each solution, both static and dynamic results are meticulously presented and elucidated through graphical representations. Overall, DQN and DDPG had a similar performance in most of the experiments, which means either one can be used. For the centralized simulation results, a comparison was made between two observation metrics the averaged normalized transmission queues' level of all stations and the probability of collision. In single-agent (i.e., centralized) static environments,

the improvement of RL approaches over BEB was 5,19% for 5 stations, 17.64% for 15 stations, 33% for 30 stations, and 47.41% for 50 stations. In single-agent dynamic environments, the improvement over BEB was 6.84% for 5 stations, 11.83% for 15 stations, 20.28% for 30 stations, and 28.43% for 50 stations. We used the collision probability as the observation metric for the decentralized simulation results. In multi-agent, (i.e., decentralized) static environments, the improvement over BEB was 5.19% for 5 stations, 17.64% for 15 stations, 16.67% for 30 stations, and 27.78% for 50 stations. Finally, in multi-agent dynamic environments, the improvement over BEB was 7.89% for 5 stations, 11.94% for 15 stations, 9.27% for 30 stations, and 8.43% for 50 stations. These graphical representations illustrate the performance enhancements achieved by DQN and DDPG compared to the traditional BEB algorithm in both static and dynamic environments. These results demonstrate the effectiveness of deep reinforcement learning algorithms in addressing complex optimization problems. DRL algorithms are, therefore, well suited to optimize network parameters such as the contention window.

Regarding the single versus multi-agent approach, the single agent had a better performance, and this can be explained through game theory. The multi-agent in a disadvantage because each station is an independent agent and attempts to maximize its rewards in every transmission, which leads to collisions. The single agent can simply make decisions considering what is best for all the stations at a given time, avoiding the complexities and potential conflicts that are present in multi-agent systems. It is concluded that all the proposed goals were, therefore, successfully achieved throughout the thesis.

Lastly, contention window optimization, though commonly associated with wireless networks, has broader applications across various technologies. In data centers, it can be adapted to manage resource conflicts between servers, improving throughput and reducing latency. In cloud computing, it helps efficiently schedule tasks among virtual machines, minimizing resource bottlenecks. For large-scale IoT systems, contention window optimization ensures the timely transmission and processing of critical data by managing communication conflicts. In industrial automation, it enhances synchronization among machines and robots, boosting productivity. Additionally, in distributed systems, it improves performance by efficiently coordinating access to shared data or tasks among multiple nodes. By adapting the principles of contention window optimization, these technologies can benefit from more efficient resource use, reduced delays, and overall better performance. Recently, the demand for services that support applications with increasingly strict performance requirements has been on the rise. Meeting this demand requires a significant boost in data transmission rates, along with

services that offer low latency, high reliability, and broad coverage. Machine learning (ML) algorithms can intelligently harness the flexibility of future communication systems, improving their spectral and energy efficiency with optimal resource allocation. The practical benefits of this approach include: a) reduced computational complexity, making it possible to implement ML algorithms in real time; b) shorter development times, as there would be no need to design specialized integrated circuits only general-purpose circuits to run pre-trained ML models; and c) lower communication system costs, resulting directly from the shorter development cycle.

5.0.2 Final Remarks

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 and decentralized 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 the superior performance of DQN and DDPG compared to BEB for both scenarios, regardless of the observation metric used and the number of stations associated with the AP. Our results show that the difference increases as the number of stations increased, with DQN and DDPG showing a 45.52% increase in throughput with 50 stations compared to BEB for the single-agent (i.e., centralized) approach and 37.16% for the multi-agent(i.e., decentralized) approach.

Additionally, the results show that DQN and DDPG had similar performances, which means that either could be used in a solution deployed in Wi-Fi APs. However, since DQN presents a lower computational complexity and lower training period, it would be the preferable choice. Furthermore, the presented results show that the performance of the network can be dramatically improved when CW is chosen based on information from the network, such as the level of the transmission queues or the probability of collision of the network. The single agent had better performance because it has the ability to unilaterally make decisions that optimize the overall throughput

in a given time. The multi-agent has a harder time coordinating what is best for the network because it is trying to maximize its own throughput.

Regarding the two considered observation metrics used for capturing the status of the network, our simulation results show that in the static scenario, they confer similar performance when applied to the DRL-based algorithms. However, in the dynamic case, the averaged normalized transmission queues level grants the algorithms higher throughput when compared to the observation based on the collision probability. These findings suggest that the averaged normalized transmission queues level provides more information to the DRL agent than the collision probability. Moreover, the centralized DRL-based solution for selecting CW values outperformed the classical 802.11 BEB algorithm, regardless of which observation metric was used. Therefore, it can be concluded that either observation metric is suitable for obtaining the status of the network and improving the CW selection, which ultimately leads to better network utilization and higher throughput.

5.0.3 Limitations and Future Work

Future work could use 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. The difference between them is that SAC is off-policy, and PPO is on-policy. In addition, a study that included these methods could be interesting because DQN and DDPG are both off-policy and use a 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.

Regarding the decentralized solution, future work could focus on making the stations cooperate by sharing information with each other. The communication between the stations is crucial for the agents to learn and converge towards a better result that optimizes the rewards for the environment instead of each individual station. Moreover, another research direction would be the assessment of the decentralized DRL-based solution using the averaged normalized transmission queues level as the observation metric since it was proven by simulation that it provides a more accurate depiction of the network status. Furthermore, the training process for the decentralized solution proposed in this work took quite a few hours and needed extensive exploration, requiring a considerable amount of computing power.

Chapter 6

Funding

This work was partially funded by CNPq (Grant Nos. 403612/2020-9, 311470/2021-1, and 403827/2021-3), by São Paulo Research Foundation (FAPESP) (Grant No. 2021/06946-0), by Minas Gerais Research Foundation (FAPEMIG) (Grant Nos. APQ-00810-21, PPE-00124-23, APQ-04523-23, APQ-05305-23, and APQ-03162-24) and by the projects XGM-FCRH-2024-1-2-1, XGM-AFCCT-2024-2-5-1 and XGM-AFCCT-2024-9-1-1 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.

References

[1] S. Barrachina-Muñoz, F. Wilhelmi, and B. Bellalta, "To overlap or not to overlap: Enabling channel bonding in high-density wlans," *Computer Networks*, vol. 152, pp. 40–53, 2019.

[2] W. Wu, Y. Liu, J. Yao, X. Fang, F. Shan, M. Yang, Z. Ling, and J. Luo, "Learning-aided client association control for high-density wlans," *Computer Networks*, vol. 212, p. 109043, 2022.

[3] S. Szott, K. Kosek-Szott, P. Gawłowicz, J. T. Gómez, B. Bellalta, A. Zubow, and 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, 2022.

[4] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Loven, A. Anttonen, A. H. Sodhro, M. M. Alam, M. Juntti *et al.*, "Machine learning meets communication networks: Current trends and future challenges," *IEEE Access*, vol. 8, pp. 223 418–223 460, 2020.

[5] Y. Yan, B. Li, M. Yang, and Z. Yan, "Esr: Enhanced spatial reuse mechanism for the next generation wlan-ieee 802.11 ax," in *IoT as a Service: 4th EAI International Conference, IoTaaS 2018, Xi'an, China, November 17–18, 2018, Proceedings 4*. Springer, 2019, pp. 265–274.

[6] T. A. Riza and D. Gunawan, "Ieee 802.11 ah network challenges supports covid-19 prevention team," in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE, 2020, pp. 73–76.

[7] Y. Lv, M. P. I. Dias, L. Ruan, E. Wong, Y. Feng, N. Jiang, and K. Qiu, "Request-based polling access: Investigation of novel wireless lan mac scheme for low-latency e-health applications," *IEEE Communications Letters*, vol. 23, no. 5, pp. 896–899, 2019.

[8] A. Al Guqhaiman, O. Akanbi, A. Aljaedi, and C. E. Chow, "A survey on mac

protocol approaches for underwater wireless sensor networks," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3916–3932, 2020.

[9] M. Gulzar, S. Kiran Kumar, M. Azhar, and S. Jabeen, "A survey on wireless channel access protocols," in *International Conference on Communications and Cyber Physical Engineering 2018*. Springer, 2023, pp. 805–814.

[10] 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.

[11] W. Auzinger, K. Obelovska, I. Dronyuk, K. Pelekh, and R. Stolyarchuk, "A continuous model for states in csma/ca-based wireless local networks derived from state transition diagrams," in *Proceedings of International Conference on Data Science and Applications*. Springer, 2022, pp. 571–579.

[12] G. Wang and Y. Qin, "Mac protocols for wireless mesh networks with multi-beam antennas: A survey," in *Future of Information and Communication Conference*. Springer, 2019, pp. 117–142.

[13] Z. Beheshtifard and M. R. Meybodi, "An adaptive channel assignment in wireless mesh network: the learning automata approach," *Computers & Electrical Engineering*, vol. 72, pp. 79–91, 2018.

[14] S.-C. Wang and A. Helmy, "Performance limits and analysis of contention-based ieee 802.11 mac," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE, 2006, pp. 418–425.

[15] M. Yazid, N. Sahki, L. Bouallouche-Medjkoune, and D. Aïssani, "Modeling and performance study of the packet fragmentation in an ieee 802.11 e-edca network over fading channel," *Multimedia Tools and Applications*, vol. 74, no. 21, pp. 9507–9527, 2015.

[16] N. Ahmed, D. De, F. A. Barbhuiya, and M. I. Hussain, "Mac protocols for ieee 802.11 ah-based internet of things: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 916–938, 2021.

[17] 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.

[18] ——, "An adaptive contention slot selection mechanism for improving the performance of ieee 802.11 dcf," *International Journal of Information and Communication Technology*, vol. 10, no. 3, pp. 318–349, 2017.

[19] R. A. da Silva and M. Nogueira, "Mac protocols for ieee 802.11 ax: Avoiding collisions on dense networks," *arXiv preprint arXiv:1611.06609*, 2016.

[20] Y. Edalat and K. Obraczka, "Dynamically tuning ieee 802.11's contention window using machine learning," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 19–26.

[21] E. F. Morales and H. J. Escalante, "A brief introduction to supervised, unsupervised, and reinforcement learning," in *Biosignal processing and classification using computational learning and intelligence*. Elsevier, 2022, pp. 111–129.

[22] M. A. El Mrabet, K. El Makkaoui, and A. Faize, "Supervised machine learning: A survey," in *2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)*. IEEE, 2021, pp. 1–10.

[23] J. Wang and F. Biljecki, "Unsupervised machine learning in urban studies: A systematic review of applications," *Cities*, vol. 129, p. 103925, 2022.

[24] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[25] H. Anouar and C. Bonnet, "Optimal constant-window backoff scheme for ieee 802.11 dcf in single-hop wireless networks under finite load conditions," *Wireless Personal Communications*, vol. 43, no. 4, pp. 1583–1602, 2007.

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

[27] H. Al-Ammal, L. A. Goldberg, and P. MacKenzie, "Binary exponential backoff is stable for high arrival rates," in *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 2000, pp. 169–180.

[28] E. Bjornson and P. Giselsson, "Two applications of deep learning in the physical layer of communication systems [lecture notes]," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 134–140, 2020.

[29] M. Kulin, T. Kazaz, E. De Poorter, and I. Moerman, "A survey on machine learning-based performance improvement of wireless networks: Phy, mac and network layer," *Electronics*, vol. 10, no. 3, p. 318, 2021.

[30] C. Silvano, D. Ielmini, F. Ferrandi, L. Fiorin, S. Curzel, L. Benini, F. Conti, A. Garofalo, C. Zambelli, E. Calore *et al.*, "A survey on deep learn-

ing hardware accelerators for heterogeneous hpc platforms," *arXiv preprint arXiv:2306.15552*, 2023.

[31] Q. Cai, C. Cui, Y. Xiong, W. Wang, Z. Xie, and M. Zhang, "A survey on deep reinforcement learning for data processing and analytics," *IEEE Transactions on Knowledge amp; Data Engineering*, vol. 35, no. 05, pp. 4446–4465, may 2023.

[32] G. Aridor, Y. Mansour, A. Slivkins, and Z. S. Wu, "Competing bandits: The perils of exploration under competition," *arXiv preprint arXiv:2007.10144*, 2020.

[33] X. Lu, B. V. Roy, V. Dwaracherla, M. Ibrahimi, I. Osband, and Z. Wen, "Reinforcement learning, bit by bit," *Foundations and Trends® in Machine Learning*, vol. 16, no. 6, pp. 733–865, 2023.

[34] S. Shekhar, A. Bansode, and A. Salim, "A comparative study of hyper-parameter optimization tools," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2021, pp. 1–6.

[35] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[36] A. Karras, C. Karras, N. Schizas, M. Avlonitis, and S. Sioutas, "Automl with bayesian optimizations for big data management," *Information*, vol. 14, no. 4, p. 223, 2023.

[37] P. Beneventano, P. Cheridito, R. Graeber, A. Jentzen, and B. Kuckuck, "Deep neural network approximation theory for high-dimensional functions," *arXiv preprint 2112.14523*, 2021.

[38] J. Zhu, F. Wu, and J. Zhao, "An overview of the action space for deep reinforcement learning," in *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, 2021, pp. 1–10.

[39] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

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

[41] 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.

[42] 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.

[43] R. Mennes, M. Claeys, F. A. P. De Figueiredo, I. Jabandžić, I. Moerman, and S. Latré, "Deep learning-based spectrum prediction collision avoidance for hybrid wireless environments," *IEEE Access*, vol. 7, pp. 45 818–45 830, 2019.

[44] M. Ahmed Ouameur, L. D. T. Anh, D. Massicotte, G. Jeon, and F. A. P. de Figueiredo, "Adversarial bandit approach for ris-aided ofdm communication," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, pp. 1–18, 2022.

[45] M. V. C. Aragão, S. B. Mafra, and F. A. P. de Figueiredo, "Otimizando o treinamento e a topologia de um decodificador de canal baseado em redes neurais," *Polar*, vol. 2, p. 1.

[46] F. Adib Yaghmaie and L. Ljung, "A crash course on reinforcement learning," *arXiv e-prints*, pp. arXiv–2103, 2021.

[47] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks," *IEEE access*, vol. 6, pp. 32 328–32 338, 2018.

[48] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, vol. 9, 2017.

[49] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *Acm computing surveys (csur)*, vol. 53, no. 2, pp. 1–33, 2020.

[50] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6g networks," *IEEE Network*, vol. 34, no. 6, pp. 272–280, 2020.

[51] P. C. Sen, M. Hajra, and M. Ghosh, "Supervised classification algorithms in machine learning: A survey and review," in *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*. Springer, 2020, pp. 99–111.

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

[53] V.-H. Nhu, A. Shirzadi, H. Shahabi, S. K. Singh, N. Al-Ansari, J. J. Clague, A. Jaafari, W. Chen, S. Miraki, J. Dou *et al.*, "Shallow landslide susceptibility mapping: A comparison between logistic model tree, logistic regression, naïve bayes tree, artificial neural network, and support vector machine algorithms," *International journal of environmental research and public health*, vol. 17, no. 8,

p. 2749, 2020.

[54] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.

[55] M. Saber, A. El Rharras, R. Saadane, A. Chehri, N. Hakem, and H. Kharraz, "Spectrum sensing for smart embedded devices in cognitive networks using machine learning algorithms," *Procedia Computer Science*, vol. 176, pp. 2404–2413, 2020.

[56] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru, "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE access*, vol. 7, pp. 158 820–158 846, 2019.

[57] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, jul 2021. [Online]. Available: https://doi.org/10.1145/3457607

[58] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? how? what to do?" in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 429–440. [Online]. Available: https://doi.org/10.1145/3468264.3468537

[59] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," *Supervised and unsupervised learning for data science*, pp. 3–21, 2020.

[60] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[61] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE transactions on knowledge and data engineering*, vol. 33, no. 4, pp. 1479–1489, 2019.

[62] V. Cohen-Addad, V. Kanade, F. Mallmann-Trenn, and C. Mathieu, "Hierarchical clustering: Objective functions and algorithms," *Journal of the ACM (JACM)*, vol. 66, no. 4, pp. 1–42, 2019.

[63] A. Abid and J. Zou, "A stochastic expectation-maximization approach to shuf-

fled linear regression," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018, pp. 470–477.

[64] H. Kour and N. Gondhi, "Machine learning techniques: A survey," in *Innovative Data Communication Technologies and Application*, J. S. Raj, A. Bashar, and S. R. J. Ramson, Eds. Cham: Springer International Publishing, 2020, pp. 266–275.

[65] J. P. Mueller and L. Massaron, *Deep Learning for dummies*. John Wiley & Sons, 2019.

[66] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application*, vol. 7, pp. 279–301, 2020.

[67] Y.-y. Z. D.-w. F. F. H. D.-s. L. Y.-m. Z. Hao-nan WANG, Ning LIU, "Deep reinforcement learning: a survey," *Frontiers of Information Technology Electronic Engineering*, vol. 21, no. 12, p. 1726, 2020. [Online]. Available: https://journal.hep.com.cn/ckcest/fitee/EN/abstract/article_28889.shtml

[68] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *J. Mach. Learn. Res.*, vol. 22, no. 1, jan 2021.

[69] T. Tiong, I. Saad, K. T. K. Teo, and H. b. Lago, "Deep reinforcement learning with robust deep deterministic policy gradient," in *2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, 2020, pp. 1–5.

[70] V. Patil, G. Ghalme, V. Nair, and Y. Narahari, "Achieving fairness in the stochastic multi-armed bandit problem," *J. Mach. Learn. Res.*, vol. 22, no. 1, jan 2021.

[71] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133 653–133 667, 2019.

[72] A. N. Burnetas and M. N. Katehakis, "Optimal adaptive policies for markov decision processes," *Mathematics of Operations Research*, vol. 22, no. 1, pp. 222–255, 1997.

[73] M. Tokic and G. Palm, "Value-difference based exploration: adaptive control between epsilon-greedy and softmax," in *Annual conference on artificial intelligence*. Springer, 2011, pp. 335–346.

[74] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp.

529–533, 2015.

[75] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 881–888.

[76] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[77] G. Alsuhli, K. Banawan, K. Attiah, A. Elezabi, K. Seddik, A. Gaber, M. Zaki, and Y. Gadallah, "Mobility load management in cellular networks: A deep reinforcement learning approach," *IEEE Transactions on Mobile Computing*, 2021.

[78] 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.

[79] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[80] T. Gonsalves and J. Upadhyay, "Integrated deep learning for self-driving robotic cars," in *Artificial Intelligence for Future Generation Robotics*. Elsevier, 2021, pp. 93–118.

[81] 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.

[82] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[83] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: A survey," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022.

[84] L. Ale, S. A. King, N. Zhang, A. R. Sattar, and J. Skandaraniyam, "D3pg: Dirichlet ddpg for task partitioning and offloading with constrained hybrid action space in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 260–19 272, 2022.

[85] H. Qi, H. Huang, Z. Hu, X. Wen, and Z. Lu, "On-demand channel bonding in heterogeneous wlans: A multi-agent deep reinforcement learning approach," *Sensors*, vol. 20, no. 10, p. 2789, 2020.

[86] B. Zhou, D. Jiang, B. Han, and T. Hayat, "Threshold dynamics and density function of a stochastic epidemic model with media coverage and mean-reverting

ornstein–uhlenbeck process," *Mathematics and Computers in Simulation*, vol. 196, pp. 15–44, 2022.

[87] 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.

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

[89] W. Wydmański and S. Szott, "Contention window optimization in ieee 802.11 ax networks with deep reinforcement learning," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

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

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

[92] A. H. Y. Abyaneh, M. Hirzallah, and M. Krunz, "Intelligent-cw: Ai-based framework for controlling contention window in wlans," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2019, pp. 1–10.

[93] A. Kumar, G. Verma, C. Rao, A. Swami, and S. Segarra, "Adaptive contention window design using deep q-learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4950–4954.

[94] I. A. Qureshi and S. Asghar, "A genetic fuzzy contention window optimization approach for ieee 802.11 wlans," *Wireless Networks*, vol. 27, no. 4, pp. 2323–2336, 2021.

[95] N. Zerguine, M. Mostefai, Z. Aliouat, and Y. Slimani, "Intelligent cw selection mechanism based on q-learning (misq)." *Ingénierie des Systèmes d Inf.*, vol. 25, no. 6, pp. 803–811, 2020.

[96] C.-H. Ke and L. Astuti, "Applying deep reinforcement learning to improve throughput and reduce collision rate in ieee 802.11 networks," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 16, no. 1, pp. 334–349, 2022.

[97] M. A. Jadoon, A. Pastore, M. Navarro, and F. Perez-Cruz, "Deep reinforce-

ment learning for random access in machine-type communication," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 2553–2558.

[98] Y. Xiao, M. Hirzallah, and M. Krunz, "Distributed resource allocation for network slicing over licensed and unlicensed bands," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2260–2274, 2018.

[99] X. Fu and E. Modiano, "Learning-num: Network utility maximization with unknown utility functions and queueing delay," in *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2021, pp. 21–30.

[100] T. K. Saini and S. C. Sharma, "Prominent unicast routing protocols for mobile ad hoc networks: Criterion, classification, and key attributes," *Ad Hoc Networks*, vol. 89, pp. 58–77, 2019.

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

[102] 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 *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2020, pp. 1–6.

[103] L. Zhang and Y.-C. Liang, "Deep reinforcement learning for multi-agent power control in heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2551–2564, 2020.

[104] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[105] 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.

[106] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[107] Z. Qu, Z. Qin, J. Wang, L. Luo, and Z. Wei, "A cooperative game theory approach to resource allocation in cognitive radio networks," in *2010 2nd IEEE*

International Conference on Information Management and Engineering. IEEE, 2010, pp. 90–93.

[108] F. Shang, X. Niu, D. He, H. Gong, and X. Luo, "Resource allocation and admission control algorithm based on non-cooperation game in wireless mesh networks," *Computer Communications*, vol. 152, pp. 63–71, 2020.