Reinforcement Learning-based Access Schemes in Cognitive Radio Networks

Ehab Maged ElGuindy
eelguindy@aucegypt.edu

AMERICAN UNIVERSITY IN CAIRO

MASTER OF SCIENCE THESIS

Reinforcement Learning-based Access Schemes in Cognitive Radio Networks

Author:
Ehab Maged ElGuindy

Supervisor:
Dr. Karim Seddik

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

in Electronics and Communications Engineering

November 29, 2020

Declaration of Authorship

I, Ehab Maged ElGuindy, declare that this thesis titled, "Reinforcement Learning-based Access Schemes in Cognitive Radio Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

AMERICAN UNIVERSITY IN CAIRO

Abstract

School of Sciences and Engineering

Electronics and Communications Engineering Department

Master of Science

Reinforcement Learning-based Access Schemes in Cognitive Radio Networks

by Ehab Maged ElGuindy

In this thesis, we propose different MAC protocols based on three Reinforcement Learning (RL) approaches, namely Q-Learning, Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG). We exploit the primary user (PU) feedback, in the form of ARQ and CQI bits, to enhance the performance of the secondary user (SU) MAC protocols. Exploiting the PU feedback information can be applied on the top of any SU sensing-based MAC protocol. Our proposed model relies on two main pillars, namely, an infinite-state Partially Observable Markov Decision Process (POMDP) to model the system dynamics besides a queuing-theoretic model for the PU queue; the states represent whether a packet is delivered or not from the PU's queue and the PU channel state. The proposed RL access schemes are meant to design the best SU's access probabilities in the absence of prior knowledge of the environment, by exploring and exploiting discrete and continuous action spaces, based on the last observed PU's feedback. The performance of the proposed schemes show better results compared to conventional methods under more realistic assumptions, which is one major advantage of our proposed MAC protocols.

Acknowledgements

I would like to thank my family for always being supportive during the time I spent at the AUC. They stood by me during all my master journey.

I want to express my deepest gratitude to my thesis advisor, Dr. Karim Seddik. He gave me technical guidance during my thesis work. Without his persistent help, the goal of this thesis would not have been realized.

Contents

List of Figures

List of Abbreviations

ACK	**A**cknowledge
ANN	**A**rtificial **N**ural **N**etwork
ARQ	**A**utomatic **R**epeat re**q**uest
CQI	**C**hannel **Q**uality **I**ndicator
CR	**C**ognitive **R**adio
CSI	**C**hannel **S**tate **I**nformation
DARPA	**D**efence **A**dvanced **R**esearch **P**rojects **A**gency
DDPG	**D**eep **D**eterministic **P**olicy **G**radient
DL	**D**eep **L**earning
DNN	**D**eep **N**eural **N**etwork
DP	**D**ynamic **P**rograming
DQN	**D**eep **Q** **N**etwork
DRL	**D**eep **R**einforcement **L**earning
DSA	**D**ynamic **S**pectrum **A**ccess
e-greedy	**e**psilon-greedy
FB	**F**eed-back
FCC	**F**ederal **C**ommunications **C**ommission
HD	**H**ard **D**etection
i.i.d.	**I**ndependently and **I**dentically **D**istributed
MC	**M**onte **C**arlo
MDP	**M**arkov **D**esicion **P**rocess
MSE	**M**ean **S**quare **E**rror
NACK	**N**egative-**A**cknowledge
No-FB	**N**o-feedback
POMDP	**P**artially **O**bservable **M**arkov **D**esicion **P**rocess
PU	**P**rimary **U**ser

RL	**R**einforcement **L**earning
SU	**S**econdary **U**ser

List of Symbols

$\mathbb{E}$	Expectation
$T(s'\|s,a)$	Transition Probability to go to s′ given state s and action a
$\Omega(o\|s',a)$	Probability of observing o given action a to end in state s'
$\mathbf{b}(s_t)$	Belief vector
$R(s,a)$	Reward function for state s and action a
$r(b,a)$	Reward function for belief vector b and action a
J^π	Future reward for policy π
η	Variable of normalization
π	Policy for the SU
π^*	Optimal policy for the SU
$V^*(b)$	optimal Value function
$r(b,a,o)$	State belief transition function
$Q(s,a)$	Q-learning for state s and action a
α	Learning rate
γ	Discount factor
$L(\theta^{(i)})$	Loss function
λ_p	Packet arrival rate
$\forall$	For all
a_s	Access probability
w	Reward of secondary user
ζ_G	Steady state probability of channel being in good state
ζ_B	Steady state probability of channel being in bad state
p_G	Probability of channel staying in good state
p_B	Probability of channel staying in bad state
μ_{sp}	SU throughput
p_d	Detection probability of the spectrum sensor

List of Publications

- **Conference paper:**
  - Ehab M. El-Guindy, Karim G. Seddik, Amr A. El-Sherif, and Tamer ElBatt, "Feedback-based Access Schemes in CR Networks: A Reinforcement Learning Approach", accepted for publication in IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, January 2021.

Chapter 1

Introduction

1.1 Cognitive radio

Radio spectrum utilization has increased to an extreme degree with the increase in demand for wireless service deployment. The ever-increasing demand for multimedia services has highlighted the frequency shortage problem more than before. This leads to higher auction rates for spectrum globally, which in turn, is passed on as a burden to the service users [1, 2].

A fixed radio spectrum is allocated to licensed users for various applications, such as TV, military, and cellular. The radio spectrum bands are controlled by a regulatory organization in the united states known as the Federal Communication Commission (FCC). The FCC found that the majority of the existing fixed available spectrum is not entirely used [2]. According to the Spectrum Policy Task Force study, only six percent of the overall fixed frequencies available are used [2]. However, the other ninety-four percent are being unutilized and causing inefficiency of the entire network. This problem has forced the FCC to develop a new scheme that allows the usage of that part of the spectrum, which is not being accessed by the licensed user in order to enhance its utilization and efficiency.

The proposed scheme that was introduced is the Cognitive Radio (CR). It was introduced by Miltola in 1999 as a combination between Software Defined Radio (SDR) and intelligent developed software-based package called cognitive engine. It enables the secondary users (SUs) to share information and communicate using that part of the spectrum, which the licensed user or primary user (PU) do not use, with the PU [1, 2, 3].

In April 2012, the FCC has adopted new rules concerning CR applications. It enabled access to unlicensed devices using an approach to white space TV database through Channels usable in the Ultra High Frequency (UHF) TV Bands. As a consequence, numerous researchers have presented several works in CR technology field, focusing primarily on its current and emerging potential applications, and the impact of its spectrum use from a technological point of view. In CR network, the concept of providing access to the unlicensed users to access the unutilized part of the spectrum, without affecting the PU is known as Dynamic Spectrum Access (DSA). DSA enables spectrum scarcity management and problems of use to be addressed [1, 2, 3].

In Feb 1958, The United States established the Defence Advanced Research Projects Agency (DARPA) for national securities. It is an independent division or agency of the pentagon. Nowadays, DARPA is encouraging and creating innovative new technologies for advanced military and civilian applications. For spectrum scarcity, DARPA launches Spectrum Collaboration Challenge (SC2) competition regularly. SC2 competition aims to solve spectrum scarcity by developing ways to share the PU's unused spectrum with the SU. SC2 combines CR with Artificial Intelligent technologies. It started in 2016, and the last one was in January 2020 [1, 2, 4].

CR is an adaptive radio technology or a communication system functions that change its performance by its own. The highest priority in CR for both PU and SU is the Quality-of-Service (QoS) when the SU accesses the free channels in a radio environment that changes with time and location. Also, the SU should not interfere with the PU services while sensing the spectrum for available channels; otherwise, it will degrade the entire network performance. The SU periodically senses the channel while utilizing it. If it senses the PU existence, it should switch to the other channel. Accordingly, the fact that spectrum resources vary with time and geographical location must be taken into account [1, 2, 3].

1.2 Cognitive radio cycle and management

CR is an intelligent radio technology that observe its surrounding environment and automatically change its parameters to maximize the utilization of the unused bands in the PU without affecting its QoS. It monitors the frequency spectrum bands in a cognitive radio cycle, acquiring the necessary information and detecting holes in the spectrum through spectrum sensing. It senses the spectrum to enable the system to determine the characteristics of the spectrum holes. Based on that, a suitable spectrum band is selected. Once the spectrum band is chosen, the user can initiate its communication over it. Moreover, CR empowers each user to periodically sense the environment to look out for any PU who is about to start its communication on the current band [2, 5].

Along with numerous advantages, CR comes with particular challenges, such as spectrum sensing technique, spectrum management, and unlicensed spectrum usage. The first CR based international standard designed to operate with a frequency band was IEEE 802.22 [2, 5, 6].

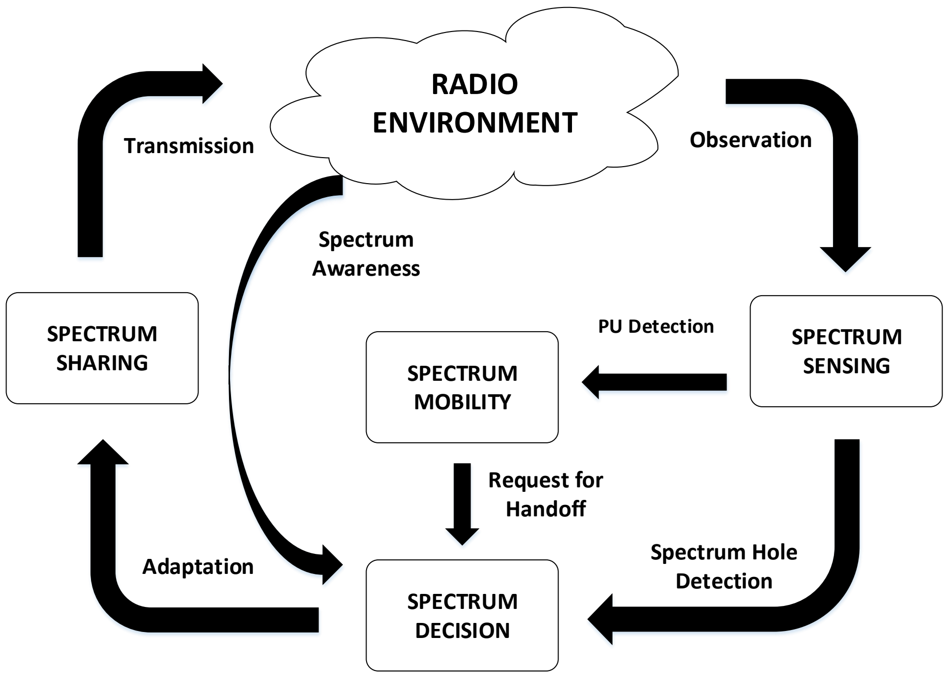

FIGURE 1.1: Cognitive Radio Cycle [2]

CR cycle and management process is divided into four main steps:

1.2.1 Spectrum sensing

It is the first and most important phase of CR process in which the SU observes the radio channel status and characteristics, such as channel interference level, PU's existence in the channel. Also, it identifies the available spectrum holes that it can use for communication. There are three kinds of spectrum sensing techniques, such as are energy detection, cyclostationary feature detection, and matched filter [2, 7]:

- Energy detection based spectrum sensing

  It is the most common way of sensing the spectrum. It has low computational and implementation complexity. It decides whether the PU is present in a channel or not from the received signal through the energy strength level. It has an advantage that no prior information about PU signal features is required. However, it is affected by noise uncertainty. It is classified into two types [2].

  1. Hard detection (HD):

     In HD, the SU compares the received energy signal level to a threshold. If this level is greater, then the PU is present, and it refrains from accessing the channel. However, if the energy level is below the threshold, it indicates the PU's absence, and it accesses the channel with probability one [8].

  2. Soft detection (SD):

     In SD, the SU compares the received energy signal level to a threshold. If this level is greater, then the PU is present, and it refrains from accessing. However. If the energy level is below the threshold, it indicates the PU's absence, and it accesses the channel with probability based on the received signal energy level [9].

- Cyclostationary feature detection based spectrum sensing:

This method involves the auto-correlation of the signal received. If the auto-correlation result is periodic, it exhibits PU's presence. However, if it is aperiodic, then it means noise, and the PU is absent. One of the advantages that it is immune to noise. However, it is complex, requires considerably more extended observation periods, and a high cost for implementation [2].

- Matched filter:

  This method detects whether the PU is active or not based on knowing prior information about the PU signal. It convolves the received signal with a mirror version of itself and shifted in time. Afterwards, it add it to a noise. This method maximizes the signal to noise ratio so that the received signal can be easily detected at different time instances [7].

1.2.2 Spectrum decision

Once the available spectrum bands are thoroughly sensed, the most suitable band for SUs transmission is chosen, that perfectly satisfies its QoS requirements. Spectrum decision process takes the following steps [2]:

- First step

  It classifies the available spectrum bands based on local observation of the SU's, PU receiver interference, and path loss.

- Second step

  A list of available channels is prepared to satisfy SU and PU activity based on the information gathered at a fusion center.

- Third step

  Lastly, these selected channels are transmitted to the SUs to prepare for communication by adjusting their transceiver parameters [2].

1.2.3 Spectrum sharing

In spectrum sharing, the SUs and PUs share the detected band between them in a coordinated way, so that they do not affect each other's communication. Spectrum Sharing can be made based on two categories.

- Spectrum sharing based on behavior of allocation
    1. Cooperative spectrum sharing:

       In this method, each user takes into account what impact its transmission would have on other user's requirements. This approach maximizes throughput and reduces interference.
    2. Non-cooperative spectrum sharing:

       In this method, each user only greedily considers how it would access the resources without taking into account other users' requirements. It has a high level of interference.
- Spectrum sharing based on access method
    1. Over-lay Spectrum Sensing:

       This approach allows the SU to transmit in available spectrum band opportunistically when the PU is absent.
    2. Under-lay Spectrum Access:

       In this method, a permissible received power of SU at the primary receiver is set as an "Interference Temperature Threshold" limit under which the SU is allowed to transmit information simultaneously with PU. Once the threshold limit is crossed, the SU transmission is stopped [2].

1.2.4 Spectrum mobility:

Spectrum mobility is when the SU switches to a separate empty band to prevent the disconnection of communication if the PU wishes to access the current channel. This channel switching is called Spectrum Handoff. There are two types of handoffs:

- Spectrum handoff based on PU arrival

  1. Reactive spectrum handoff:

     The SU senses the target channel on demand, then switches to that channel on request. This process causes delays since the switching of channels takes place after the handover action.

  2. Proactive spectrum handoff:

     In this scenario, the users of the CR track a PU's activity consistently, so the SU conducts the handoff and spectrum sensing as soon as it detects the behavior of the PU's arrival. It has no latency, yet due to the inaccurate PU estimation of arrival, it may perform a wrong handoff.

- Spectrum handoff based on guard channels

  In this case, as long as there is no interference caused to adjacent users, the SU can use guard channels for communication when no vacant band is found [2].

1.3 PU reverse traffic

In this thesis, we are interested in the design of the SU access schemes based on the PU feedbacks. In the CR, the PU transmitter-receiver feedback message plays a significant rule as it offers the PU transmitter an indicator of the quality of transmission at the receiver. Furthermore, the SU can use this feedback to estimate the PU activity. There are two different PU feedbacks which can be used by the SU:

- Automatic Repeat Request (ARQ)

  ARQ originally referred to controlling error in data transmission by sending back acknowledgment (ACK) when a correct data is received or negative acknowledgment (NACK) when a false data is received. In this process, the error in the received packet is detected by the recipient with error detection code. In case when there is detection from the receiver for no error in the data received, it notifies the sender with (ACK). Whereas, when there is detection for a code error, it drops the data packet and sends (NACK) to the sender; this (NACK) serves as a request to attempt retransmission of the data. These (ACK/NACK) signals are a short text, which indicates the correctness in the receipt of the received data. The sender also triggers retransmission in case of absence of (ACK/NACK) in the defined time interval. The sender waits for the (ACK/NACK) till it times out and then retransmits the data up to a certain number of retrials. Similarly, it can be used for detecting the PU activity. In the ARQ based system, we have three possible PU ARQ feedback states: ACKs, NACKs, and No-Feedback (No-FB). An ACK denotes a successful transmission, a NACK denotes a failed transmission, and No-FB means that the PU did not attempt any transmission in the last time slot. In an ideal, or perfect-sensing system, the SU will have complete knowledge of the PU state if it is active or idle in each time slot. On the contrary, in our system, we assume that The SU gets only feedback from the PU information [10].

- Channel Quality Indicator (CQI)

  CQI indicates how good or bad a communication channel is. The SU decides to access the channel through the CQI feedback. If the CQI feedback from the PU is bad, then it indicates that the PU is not going to transmit considering the bad channel state; hence, the SU can access the channel with probability one. Whereas, if the CQI feedback is good, it indicates that the PU may or may not be accessing the channel depending on the arrival rate. In practice, there can be more than these two (good/bad) CQI levels. For example, LTE networks have 16 different CQI levels, where each level corresponds to a coding and modulation selection. When the CQI level is minimum, the transmitter chooses not to transmit, considering the channel's bad state [11].

1.4 Cognitive radio challenges

The study and analysis of this extraordinary technology will be among the most exciting and compelling findings of the scientific world in the 21st century. Several researchers targeted CR systems (CRS) due to its application in different disciplinary areas and acquired near-optimal results. Nevertheless, CRS has multiple operating modes and complex architectural design and has particular challenges, such as decision making, cross-layers design, security breaches, learning process, and spectrum sensing techniques. However, The most significant aspect of CR networks and their implementation is spectrum sensing. Spectrum sensing opportunistically exploits the underused resources, such that the available resources can be used efficiently [5, 12]. Moreover, learning in CR aims only on collecting and analyzing data.

CR's main idea is its cycle. As the cycle repeats itself, the radio should be able to learn from experience actions. However, that does not happen as it misses the intelligence part of not repeating the faults. Taking advantage of the AI technology, CR can learn from past actions. Reinforcement Learning (RL) model-free algorithms are sub-fields of AI that learn and adapt to the entirely unknown system dynamics. Besides, Machine Learning (ML), which is a sub-field of AI, is used with RL to enhance the action decision making [12, 13, 14].

1.5 Summary of latest work on MAC protocols access Schemes for Cognitive Radio Systems

The evolution of CR by authorizing the SUs to exploit the underutilized spectrum of the PUs is essential for solving the scarcity problem of the unused frequencies. However, the SUs access is constrained by not affecting the PUs network. QoS level is always guaranteed for the PUs in the presence of SUs. SUs Medium Access Control (MAC) schemes have attracted much interest over the last few years [15].

Many papers have focused on using the PU's feedback information to devise better SU's MAC schemes. Eswaran et al. in [16] have considered the use of ARQ messages received by the SU, to perceive the packet rate achieved by the PU; the aim is to maximize the secondary throughput while ensuring a minimal PU packet rate. Furthermore, based on the primary link feedback, secondary power control is explored in [17]. While maintaining particular PU QoS requirements, the goal was to maximize the SU utility in a distributed scheme. Moreover, a PU retransmission based error control scheme is studied in [18] for the sake of designing an optimal transmission policy for the SU. Based on the PU retransmission packet state, the SU determines its transmission strategy.

Seddik et al. presented in [19] a secondary access scheme that exploits the ARQ feedback; the SU refrains from accessing the channel upon hearing a NACK feedback (FB) from the primary receiver to allow for collision-free retransmission and avoid certain collisions with the PU. The scheme is shown to achieve higher secondary user throughput while guaranteeing the PU QoS constraint. Beside soft-energy sensing, the authors in [20] proposed PU feedback information for designing the SUs' access scheme under a PU stability constraint. Afterward, the work in [21] introduced a POMDP framework to design a SUs' MAC protocol exploiting the history of the PU feedback bits. Intuitively, a greedy algorithm was proposed to simplify the solution for the proposed POMDP problem. The greedy algorithm proposed an approximation threshold number for the SU to take the access decision. The greedy algorithm's goal was to maximize the instantaneous SU reward. However, the work assumed perfect knowledge of the PU arrival rate, which might not be available in many real-life scenarios.

Moreover, there has been another direction for designing SUs' access schemes by exploiting the CQI feedback [8]. Attalla et al. in [8] and [22] implemented the same system with two models; one with single SU and PU and other with multiple SUs and PUs. Their experiment showed that SUs only use the CQI feedback knowledge from the PUs. In combination with soft energy sensing, the system showed gains in performance. The

SU is accessing the primary channel with chosen access probabilities to optimize the SU rate of service while maintaining the QoS for the PUs. A multidimensional Markov chain was used to design the proposed system. The estimated packet delays and closed-form expressions of SUs were calculated. Besides, the selection of access probabilities of SUs is formulated as a restricted problem of optimization. Regarding the SU service rate and the initial delay of the user, the implemented scheme's output is shown to be superior concerning others where SUs do not take advantage of CQI feedback data from PUs.

Besides, the work in [23] examined the design of hybrid schemes that exploit both ARQ and CQI feedback with hard and soft energy sensing techniques, which enables better monitoring of the activities of the PUs. Comparing hard energy-sensing in [24], where the expected PU state is busy or idle, the significant advantage of using soft energy sensing in [23] is helping the SUs to make better decisions in accessing that rely on their confidence in the existence of PUs. A Markov chain of three-dimensional homogeneous quasi birth and death was studied to determine the stable-state distribution of the queue. The optimization problem of increasing the secondary network throughput depending on the performance limitations of the PU queues. ARQ allows preventing PU collisions when PUs retransmit failed packets, and CQI allows SU to transmit when PUs refrain from the transmission.

In this thesis, we focus on the design of SUs' access schemes exploiting the primary network available feedback information in the form of ARQ and CQI feedback bits. The problem of the design of the SUs' access scheme is formulated as a POMDP. We focus on the use of different model-free RL approaches, namely Q-Learning, Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG), to design SUs' access schemes (i.e., solve the formulated POMDP problem) that require minimal knowledge about the PU parameters. The proposed RL based schemes are shown to achieve the same performance of the previously proposed schemes, which have some impractical, or hard to achieve, assumptions like knowing the

PUs' arrival rate. Our proposed schemes would allow for online implementations that can adapt to variations in the primary network, which is not the case with the previously proposed "static" algorithms.

1.6 Thesis organization

This thesis presents a contribution to the access schemes of the cognitive MAC protocol layer by using Reinforcement Learning algorithms to explore the environment and exploit different feedbacks information from the primary user. The thesis is organized as follows:

- Chapter (2) presents background on Reinforcement Learning
- Chapter (3) discusses the access techniques utilizing solely ARQ or CQI feedback using RL algorithms Q-learning, DQN, and DDPG.
- Chapter (4), discusses the access techniques utilizing both ARQ and CQI, namely, the hybrid system, using RL algorithms Q-learning, and DDPG.
- Chapter (5) concludes the thesis and presents directions for future work.

1.7 Thesis contributions

In the following, we list the thesis's major contributions:

1. We propose a SU access scheme that exploits the PU ARQ and/or CQI feedback (on the top of any spectrum access scheme). We formulate the problem of the SU access decision as a POMDP and use different RL algorithms to approach solving this problem.
2. We show that our proposed RL based approach achieves results that are comparable to other previously proposed algorithms while

not requiring minimal information about the PU; unlike the previous work which has made some unrealistic assumptions (like the knowledge of the PU arrival rate, probability of detection, probability of false alarm, etc).

3. We achieved minor throughput gain by using DDPG RL algorithm that takes better access probabilities by using continuous action space rather than a discrete one.

Chapter 2

Background on Reinforcement Learning

This chapter provides a background on a branch of Machine Learning (ML), which is RL. RL started with model-based algorithms such as Dynamic Programming (DP). Afterward, a model-free algorithm was introduced, which is the Temporal Difference (TD). TD consists of two algorithms, Sarsa and Q-learning. For better performance, algorithms such as DQN and DDPG were introduced, which combines deep learning with RL .

2.1 Machine learning classifications

ML is a branch of AI that studies algorithms and develop an intelligent system based on the human mind. They are explicitly programmed to enable the learning skills of the program. It is a concept and not language-dependent. Its algorithm has three classified classes [25]:

- Supervised learning:

  The machine is trained to predict the outcome based on labeled data. The model planned for a solution may be a classification or a regression problem. Some of its applications are image classification, object detection, and augmentation [25].

- Unsupervised learning:

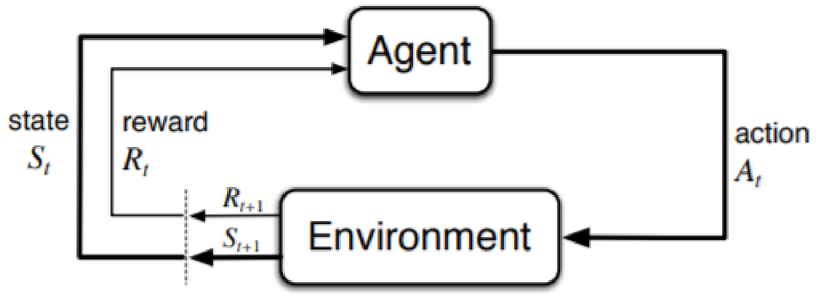

FIGURE 2.1: Markov Decision Process [25]

The machine is trained to find patterns in the data, based on unlabeled data. The model could be solved by clustering. One of its applications is the difference between genes in the DNA micro-array data [25].

- Reinforcement learning:

  RL is the agent or machine that is like humans or animals in the way it learns. It learns by trial and error interactions with the environment through mapping state to action and receiving a reward without the need for prior knowledge about the dynamics of the environment. After a short time, it becomes an expert with the best policy that maximizes long-term rewards. Some of its applications are Robotics, Telecommunication, and control theory [25, 26, 27, 28].

2.2 MDP model

The dynamics of the environment in RL can be modeled as a Markov decision process (MDP) as shown in Fig. 2.1.

There is an interaction between two entities; The agent, which is the SU, and the environment, the surroundings, or everything the agent interacts with. Through sequential discrete time steps $t = 0, 1, 2, 3$, the agent forms a sequence of interactions. For example, at time step t, it observes a state S_t from the environment. Based on that, it perform action A_t. Afterward, the environment replies with a reward in the next time step R_{t+1} and a new state S_{t+1} and so on. At the end, the MDP forms a group of sequential observations, actions, and rewards.

R_t and S_t are random variables that depend only on the previous state and action due to Markovian properties as show in equation (2.1)

$$P(s'|s,a) = p(s_{t=s'}|s_{t-1=s,A_{t-1=a}}) = \sum_{r \in R} P(s', r|s, a) \tag{2.1}$$

The agent's main goal is to increase the amount of rewards it receives R_{t+1}, and $R_{t+2}, ...$ which can be presented by G_t from a specific actions taken in a states observed. It can be presented as in equation (2.2)

$$G_t = R_{t+1} + \gamma \times R_{t+2} + \gamma^2 \times R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k \times R_{t+k+1} \tag{2.2}$$

where γ is the discount rate, and it takes values from zero to one. The main function of $Gamma$ is to prevent G_t from becoming infinity in continuous tasks. If $Gamma$ is set to zero, that means the agent cares more about the current step's reward. However, if it is one, it means that the agent gives the same value for the reward of the current to the last step [25].

2.2.1 Policy function

The policy maps states to actions. It can be categorized into three types, Deterministic, Stochastic, and Random policy. Besides, there are on-policy and off-policy algorithms. In stochastic policy, the agent will be following a policy π under a probability of selecting specific action given a state $a = \pi(a|s)$. In deterministic policy, the agent will select one action for a given state $a = \pi(s)$. In random policy, the agent randomly selects any action to maximize the rewards $a = rand(A)$.

In on-policy algorithms, the value of the action taken by the agent is considered, while in off-policy algorithms, the value may or may not be of the action taken by the agent [28, 29].

- Exploration vs Exploitation

  The main goal for the agent in both methods is to find the best policy that maximizes the reward gained from the environment. It is a trade-off between selecting the best action or trying a new one. Exploitation is when the agent has information and acts greedily upon it to maximize the return reward. However, during exploration, the agent chooses to randomly explore all possible actions that can result in a better reward. There is a technique called ϵ-greedy method, which means that the agent will behave randomly with probability ϵ and act greedily with probability $1 - \epsilon$. This ϵ value can be adjusted based on the demand for exploration and exploitation in the environment [28].

- Episodic task vs. Continuous task

  In some cases, the environment has a terminal state. A terminal state is defined as the final state. For example, in a game, the terminal states would be a win, lose, or draw. In a continuous task, there are no terminal states. For example, when the agent is learning how to drive, it does not stop learning. However, in episodic tasks, there are terminal states. Therefore, in episodic tasks, the episode ends when the agent reaches a terminal state [25, 28].

2.2.2 Value function and Action-Value function

The value function is the measure of how good the agent is doing in each state by following a policy π and continuing to play to the end before the step is taken as shown in equation (2.3)

$$v_\pi(s) = E[R_t|s_t = s] \tag{2.3}$$

Adding Bellman equation to solve the MDP and find a relation between the current state and its successors as in equation (2.4)

$$v_\pi(s) = \sum_a \pi(a|s) \times \sum_{s',r} p(s', r|s, a)[r + \gamma \times v_\pi(s')]) \tag{2.4}$$

The optimal value function is the goal to reach for convergence which is the maximum value function achieved as in equation (2.5)

$$v_*(s) = max_\pi v_\pi(s) \tag{2.5}$$

The action-value function is the measure of how the agent is doing well after each action in each state S by following a policy π and continuing to play to the end as shown in equation (2.6)

$$q_\pi(s,a) = E[R_t | s_t = s, a_{t=a}] \tag{2.6}$$

Adding Bellman equation to solve the MDP and find a relation between the current action-value pair and its successors as in equation (2.7)

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a'))] \tag{2.7}$$

The optimal action-value function is the goal to reach for convergence which is the maximum action value achieved as in equation (2.5)

$$q_*(s,a) = max_\pi q_\pi(s,a) \tag{2.8}$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \times max_{a'} q_\pi(s',a'))] \tag{2.9}$$

[26]

2.3 Model-based vs Model-Free algorithm

There are two different ways to optimize the model's policy to give the highest reward, which is model-based and model-free. The main difference

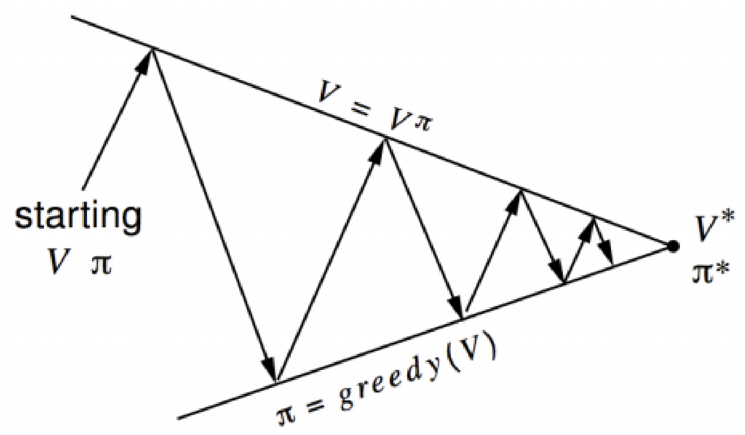

FIGURE 2.2: Policy Iteration

is that in model-based, the states' transition probabilities and the rewards are known and can be calculated before the game to estimate the policy. However, model-free method assumes no prior information of the model and follow a random policy until reaching the optimal and get the maximum reward. For model-based methods, there is DP algorithm that acquires transition state and complete prior information about the model. On the other hand, for model-free methods, there are Monte Carlo and Temporal Difference (TD) Algorithms that require no complete prior knowledge of the environment [25].

2.3.1 Dynamic Programming (DP) (Model-based) algorithm

After inheriting Bellman equation in the value and action-value function, we still can not mathematically solve the RL problem with a large state number. For that reason, DP was introduced in 1957 to find the optimal value policy function iteratively. We assume that we know the environment's states transition matrix and rewards function. It consists of policy Evaluation, Improvement, and Iteration.
Policy evaluation is about calculating the value functions and the updates as in equation (2.10)

$$v_\pi(s) = \sum_a \pi(a|s) \times \sum_{s',r} p(s',r|s,a)[r + \gamma \times v_\pi(s')]) \qquad (2.10)$$

Policy Improvement is about calculating the maximum action value function as shown in equation (2.11)

$$\begin{aligned} \pi^{'}(s) &= argmax_a q_\pi(s,a) \\ &= argmax_a E[R_{t+1} + \gamma v_\pi(S_{t+1}|S_{t=s,A_{t=a}}) \\ &= argmax_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]) \end{aligned} \tag{2.11}$$

Policy iteration is about to keep iterating until satisfying Bellman equation and reaching the optimal policy and value, which is convergence, as shown in Fig. 2.2 [25].

2.3.2 Monte Carlo (MC) (Model-Free) algorithm

Unlike DP, MC is a model-free algorithm that works without any prior information about the environment dynamics. However, it assumes that the agent has a terminal state and endpoint regardless of the action selected. It is considered to be repeated trials of random paths that consist of states, actions, and rewards by the interactions with the environment. The action is non-deterministic or random. The method calculates the average return ($v(s)$) for all states after reaching a final state. As soon as the average returns for all states become constant, the agent stops as it reached convergence. Therefore, all states inside the model have been visited or explored [25].

2.3.3 Temporal Difference (TD) (Model-Free) algorithm

TD is a model-free algorithm that combines MC and DP. It learns as MC. Nevertheless, it updates its estimate each time step as DP. It experiences the next states and received rewards by using the value and action-value function. The action is deterministic. It consists of two known algorithms [26]:

- Sarsa

It is an on-policy method where in the Q-values are updated every time step. When the agent moves from time step t to $t+1$, the five elements used in the update equation are $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. These five elements what gives SARSA its name. The update equation for SARSA is showing in equation (2.12).

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{2.12}$$

- Q-learning

  Q-learning is an off-policy model-free algorithm that started in 1989. It was an evolution for RL. Its main objective is to learn the policy, enabling the agent to take action under specific situations. Also, it stores data in tables and infers the action to improve the policy. In tables, we keep states, and for each state, we have available sets of actions. The update equation (2.13) is similar to that of SARSA, except in Q-learning, the maximum overall Q-values corresponding to s' and a' is selected [26].

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{2.13}$$

2.4 Deep reinforcement learning

Machine Learning was integrated with RL due to the demand to solve the high-dimensional input system space problems. Also, due to the neural network's efficiency in extracting important features and finding the non-linear relation between the input and the output. Multi-layer perceptrons were introduced. Its main function is to approximates the action-value function $q(s,a,\theta)$ as well as the policy function $\pi(a|s,\theta)$ [26, 28].

2.4.1 Deep learning

Deep Learning is a subfield of machine learning. It beat humans in games as it is much more efficient to read the environment, take actions, and improve

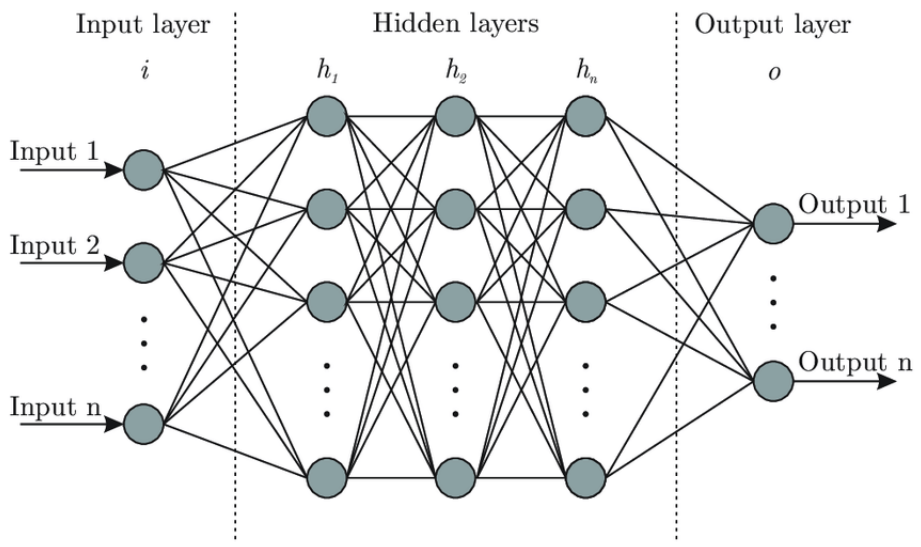

FIGURE 2.3: An Artificial Neural Network (ANN)

by playing. It is based on neural networks called Artificial Neural Network (ANN), but with more than two hidden layers. Neural network is a group of nodes (or neurons), fully connected in columns as layers, as shown in Fig. 2.3. It consists of three types of layers: the input, the hidden, and the output layer. First, the input layer takes the features, which can be the observation states and actions. Second, there can be more than one hidden layer in a fully connected neural network. Lastly, the third layer is for the output layer, which result can be a classification or regression problem to define the relation between the predicted data and the ground truth. There are forward and backward propagation in the neural network. In forward propagation, the input nodes are multiplied by weights, which are summed in each neuron of all hidden layers. Afterwards, it is added to the neuron's bias and passes through activation function function. This activation functions' main goal is to set a threshold for the neuron to be activated or not. One of the famous examples of activation functions is the sigmoid function. Finally, The output layer receives the multiplication and the fired neurons of the hidden [26, 30, 31].

For measuring how good the model is, a loss function is calculated. It is the difference between the actual value and the predicted one. One of the loss functions is called the mean square error (MSE). The more the loss function's value is small, the more the agent is getting close to the optimum

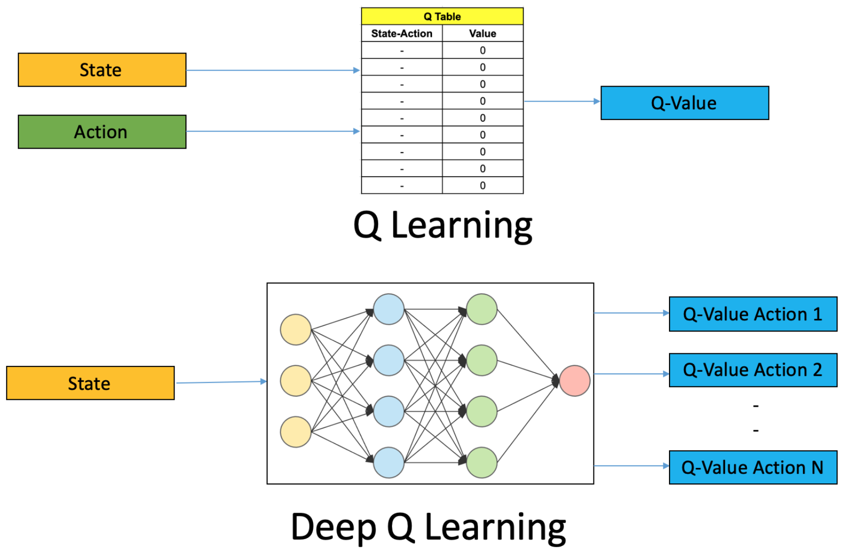

FIGURE 2.4: Deep Q-Network

value. In the back propagation, all the weights are updated by taking its partial derivatives as part of the optimization method. There are different types of weights gradient descent methods to decrease the error function. Finally, we keep iterating forward and backward until convergence and reaching the global minimum point [26, 30, 31].

2.4.2 Deep Q-Network (DQN)

Q-learning algorithm was unable to handle large-state input space, due to the demand for high-dimensional input state problems. For that reason, DNN was inherited with RL. Neural network is a non-linear approximation function that is trained with $q(\theta)$ parameters. Besides, neural network can better estimate the Q values due to its better feature extraction. That led to speed increase, and helped in removing hardware limitations in case of large state-input dimensional. DQN makes use of both Q-learning as well as neural network. It uses neural networks instead of Q-tables to store and show the state-action value, as shown in Fig. 2.4 [26, 32, 33].

$$L(\theta^Q) = [Y_t - Q(s_t, a_{t|\theta^Q})]^2 \tag{2.14}$$

$$Y_t = [r_{t+1} + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\theta^Q)] \tag{2.15}$$

2.4.3 Deep Deterministic Policy Gradient (DDPG)

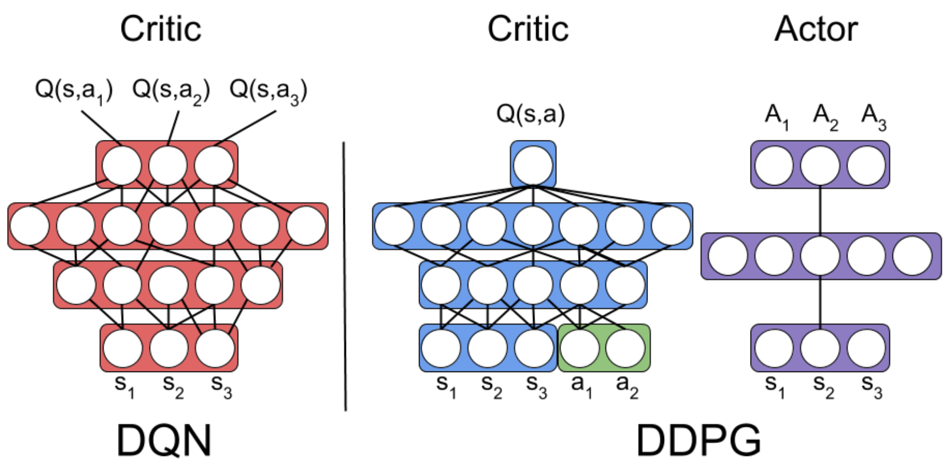

FIGURE 2.5: Deep Deterministic Policy Gradient

DQN achieved great results in high dimensional state space applications. However, its action space is discrete, which could not be the best choice for exploration in some applications. RL is all about exploring states and actions. Continuous action space is crucial for applications like angular wheel drive, Telecommunications, and military applications, etc.

In order to make the action space continuous, the policy should be in a separate network, as shown in Fig. 2.5. DDPG combines both the value function and the policy function. The policy function is for the actor network to make it continuous and regression problem instead of classification. While, the value function is for the critic network for classification problem [26, 27, 34].

Chapter 3

Automatic Repeat Request (ARQ) and Channel Quality Indicator (CQI) based Access Schemes

In this chapter, a CR system is designed whereby the SU utilizes the ARQ or CQI reverse traffic of the PU. Based on the SU spectrum sensing outcome and PU feedback, it randomly accesses the PU channel. Also, we consider the hard detection (HD) sensing technique in the CQI feedback model. RL algorithms that are used to learn policies for access probabilities to solve the proposed POMDP problem are, Q-Learning, DQN, and DDPG.

3.1 Automatic Repeat Request (ARQ) based access scheme

In this feedback system, whenever a packet arrives in the PU's queue, the PU transmits to the receiver. The SU has access to feedback between the primary transmitter and receiver. Observing No-FB from the SU means that the PU's queue is empty, while ACK indicates a successful packet was sent from the primary TX to the primary RX. Consequently, the SU access the channel based on the packet arrival rate in the PU's queue with an access probability a_s. On the other hand, upon receiving NACK feedback, it means that there was a collision between the PU and the SU's packets, and

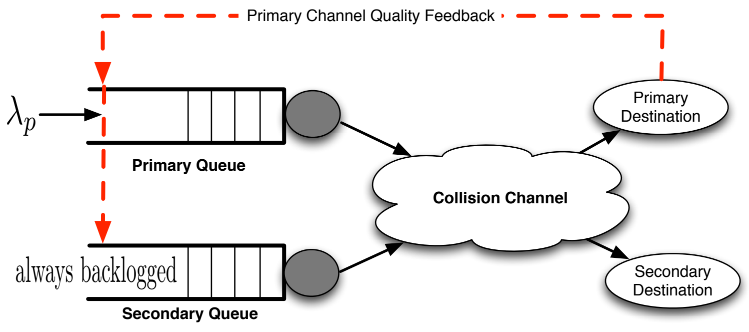

FIGURE 3.1: The System model

the PU's packet was not delivered. So, the SU refrains from accessing as the PU TX will be retransmitting its last unsuccessful packet attempt in the next time slot [21].

3.1.1 System model

A time-slotted system is considered in all our models, consisting of one PU and one SU, as shown in Fig. 3.1. The PU hosts an infinite buffer to store incoming packets. The PU average arrival rate is λ_p packets per time slot; the arrival process is assumed to be a Bernoulli process with independent and identically distributed arrivals. The packet's transmission time is assumed to fit exactly within the duration of one slot. Therefore, λ_p will assume values in the range $0 \leq \lambda_p \leq 1$. Otherwise, the stability of the PU queue will not be attained. Also, it is assumed that the SU's queue is always backlogged, i.e., the SU will always have a packet to transmit. The SU employs a random access scheme with access probability $a_s(.)$, i.e., a slotted ALOHA access scheme is adopted. The access probability will be adapted depending on the SU estimate of the PU activity (PU state). In our proposed model, the access probability will be a function of the PU feedback, whether ARQ and/or CQI feedback states as will be discussed later. In addition, we assume a collision channel model. If either the PU or the SU transmits in any time slot, this will result in a successful transmission. However, packets can only be lost in communication in the case of concurrent PU and SU transmissions. In this case, a collision is declared, and all the packets involved

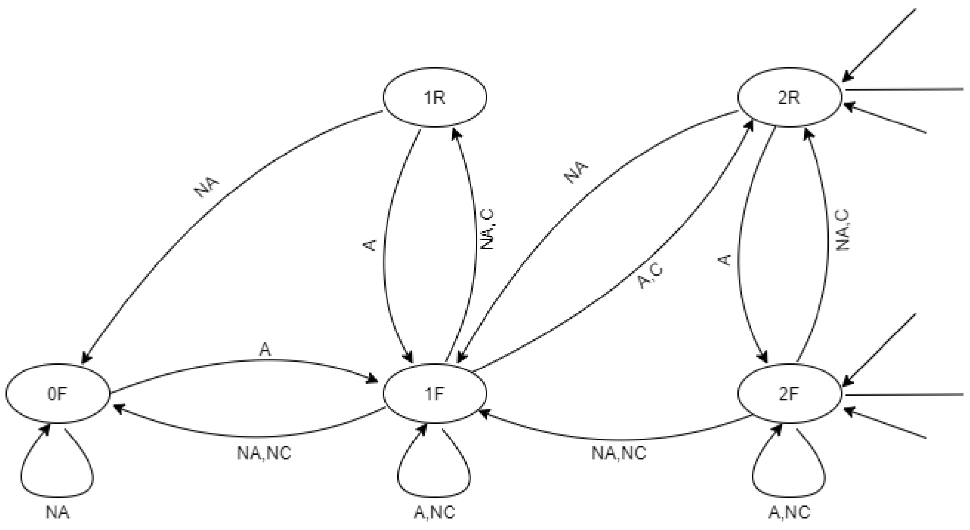

FIGURE 3.2: Two-dimensional MC model for the PU queue

will be lost. The PU receiver will send a NACK feedback for the PU to re-transmit its packet. We also assume that all feedback information is always received correctly at the receiving nodes as these feedback bits usually are well protected by secure channel codes [21].

3.1.2 The PU queue model

We model the system dynamics between the PU TX and RX as a Two-dimensional MC model by exploiting the ARQ feedback, as shown in Fig. 3.2. Inside each circle there are two parameters, which are the state space, the number of packets in the queue, and the i_F's and i_R's states. The subscript F refers to the first transmission, while the subscript R indicates the PU re-transmission state, after receiving a NACK feedback. The arrows represent the transition from one state to another. For example, going from $0F$ to $1F$ can happen if there is arrival (A). Also, going from $1F$ to $0F$ happens if there is no arrival (NA) and no collision (NC) [21].

3.1.3 POMDP environment framework

A POMDP is defined by the tuples $(\mathcal{S},\ \mathcal{A},\ \mathcal{O},\ T,\ \Omega,\ R)$, where the set $\mathcal{A}$ denotes the set of SU actions (which correspond to different SU access probabilities). The set $\mathcal{S}$ denotes the PU Markov chain states $S = \{\{i_F\}, \{j_R\}\}$,

$i = 0, 1, \cdots$ and $j = 1, 2, \cdots$. The set $\mathcal{O}$ defines the observations set that is presented by $\mathcal{O} = \{\text{ACK, NACK, No-FB}\}$. The function $T(.)$ denotes the transition probabilities function, where $T(s'|s, a)$ indicates the likelihood to go from state s to state s' given action a. To better illustrate it and following the formulation in [21], which is reproduced here for convenience, we give few examples below for $T(s'|s, a)$, for different values of s, s' and a.

$$
\begin{aligned}
&T(i_R|j_F,\ \text{no access}) = 0,\ \forall i,\ j\\
&T(i_F|j_F,\ \text{access}) \quad = 0,\ \forall i,\ j \neq 0\\
&T(1_F|0_F,\ \text{access}) \quad = \lambda_p,\\
&T(1_F|0_F,\ \text{no access}) \quad = \lambda_p,\\
&T(i_R|j_F,\ \text{access}) \quad = \begin{cases} 1-\lambda_p & \text{if } i = j, j \neq 0 \\ \lambda_p & \text{if } i = j+1, j \neq 0 \\ 0 & \text{otherwise,} \end{cases}\\
&T(i_F|j_R,\ \text{no access}) = \begin{cases} 1-\lambda_p & \text{if } i = j-1 \\ \lambda_p & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{3.1}
$$

The function $\Omega(o|s', a)$ denotes the probability of observing o given that action a is applied to result in state s'. It can be estimated as [1]

$$\Omega(o|s' = i_F,\ \text{no access})$$
$$= \begin{cases} 0 & o = \text{NACK and } \forall i_F \\ P_{\text{ACK}}(0_F) & o = \text{ACK, } i_F = 0 \\ P_{\text{No-FB}}(0_F) = 1 - P_{\text{ACK}}(0_F) & o = \text{No-FB, } i_F = 0 \\ P_{\text{ACK}}(1_F) & o = \text{ACK, } i_F = 1 \\ P_{\text{No-FB}}(1_F) = 1 - P_{\text{ACK}}(1_F) & o = \text{No-FB, } i_F = 1 \\ 1 & o = \text{ACK, } i_F \geq 2 \\ 0 & o = \text{No-FB, } i_F \geq 2 \end{cases} . \tag{3.2}$$

If we begin with a certain vector of belief $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the index of time, then the new vector of belief is given after the action a_t observing some o_{t+1}. However, the formula values will not be affected since, the reward of SU will always be zero in the no-access state (as described later) [21].

$$\Omega(o|i_F,\ \text{access}) = \begin{cases} 0 & \forall o \text{ and } i_F \geq 2 \\ 1 & o = \text{No-FB, } i_F = 0, 1 \\ 0 & o = \text{ACK or NACK, } i_F = 0, 1 \end{cases} \tag{3.3}$$

$$\Omega(o|i_R,\ \text{no access}) = 0 \quad \forall o \tag{3.4}$$

[1]By abuse of notations, we set $\Pr(A|B) = 0$ if $\Pr(B) = 0$ (for example we set $\Omega(o|i_R,\ \text{no access}) = 0$ since $\Pr(i_R,\ \text{no access}) = 0$ under our collision system model assumption).

$$\Omega(o|i_R, \text{access}) = \begin{cases} 1 & o = \text{NACK} \\ 0 & \text{otherwise} \end{cases}. \tag{3.5}$$

The function $R(.)$ denotes the reward function calculated as follows.

$$R(s,a) = \begin{cases} r & a = \text{access}, s = 0_F \\ -1 & a = \text{no access}, \forall s = 0_F \\ 1 & a = \text{no access}, \forall s \neq 0_F \\ -1 & a = \text{access}, s \neq 0_F \end{cases}. \tag{3.6}$$

It is an immediate reward that the SU has earned to take a specific action and reach a new state. If the queue of the PU is empty, i.e., $s = 0_F$ and the SU accessed the channel, it will gain a positive reward. However, in this same case and if the SU does not access the channel, it receives a penalty due to the lost transmission opportunity. Moreover, if the queue is not empty and the SU accessed, it also receives a penalty. On the other hand, if the SU does not access the channel, it receives a positive reward for avoiding a certain collision with the PU [21].

The belief vector is given by $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the time index. Its the probability distribution over states based on the history at time t. After taking an action a_t and observing some o_{t+1}, the new belief for some state s_{t+1} at time $(t+1)$ is given by

$$b(s_{t+1}) = \eta\Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b(s_t), \tag{3.7}$$

As η is considered to be the variable of normalization shown by

$$\eta = \frac{1}{\sum_{s_{t+1} \in \mathcal{S}} \Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b(s_t)}.$$

[21]

3.1.4 POMDP MAC Policy

This section describes the mapping of the belief vector to the action space. This mapping is affected by the reward of the current state and the expected reward in the following states, which is governed by the dynamics of the Markov chain. This is attributed to the fact that the belief vector in the next time instant will be affected by the present action. It is possible to model the MAC policy as a Markov decision process based on the belief vectors (belief MDP). Based on a belief vector $\mathbf{b}$ and an action a, the expected reward is given by

$$r(\mathbf{b}, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a). \tag{3.8}$$

For any belief vector $\mathbf{b}$, the SU access policy π is defined by an action $a_\pi = \pi(\mathbf{b})$[2]. Over an infinite horizon, the accumulated reward is the objective function to be maximized. Starting with a belief vector $\mathbf{b}_0$, the estimated reward for policy π is given by

$$J^\pi(\mathbf{b}_0) = \sum_{t=0}^{\infty} \gamma^t r(\mathbf{b}_t, a_t) = \sum_{t=0}^{\infty} \gamma^t E\Big[R(s_t, a_t) \mid \mathbf{b}_0, \pi\Big] \tag{3.9}$$

where $0 \leq \gamma < 1$ is a discount factor. The optimal policy π^* is given by

$$\pi^* = \underset{\pi}{\text{argmax}} \;\; J^\pi(\mathbf{b}_0) \tag{3.10}$$

where $\mathbf{b}_0$ is the initial belief vector as defined above [21].

For each belief state, the maximum expected reward value specifies the optimal policy, π^*. It is closely modeled by the best value function V^*, which is the solution for the following Bellman equation

$$V^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \Big[r(\mathbf{b}, a) + \gamma \sum_{o \in \mathcal{O}} \Omega(o \mid \mathbf{b}, a) V^*(\tau(\mathbf{b}, a, o))\Big], \tag{3.11}$$

where $\tau(\cdot, \cdot, \cdot)$ is the belief state transition function [21].

[2]Note the optimal policy can be defined as a probability measure over the action space that is a function of the belief vector, i.e., the policy defines the probability for each action under a certain belief vector.

TABLE 3.1: ARQ Feedback

s(t-1)	q(t) = length of queue	a(t+1)	S(t+1)	r(t+1)	q(t+1) because of action	q(t+2) because of arrival
0 = No-FB	0	0 = no access	0 = No-FB	-1	0	depends on the arrival rate
0 = No-FB	0	1 = access	0 = No-FB	1	0	
0 = No-FB	Greater than or equal to 1	0 = no access	1 = ACK	1	q = q-1	
0 = No-FB	Greater than or equal to 1	1 = access	2 = NACK	-1	q(because it has to send the same packet again	
1 = ACK	0	0 = no access	0 = No-FB	-1	0	
1 = ACK	0	1 = access	0 = No-FB	1	0	
1 = ACK	Greater than or equal to 1	0 = no access	1 = ACK	1	q = q-1	
1 = ACK	Greater than or equal to 1	1 = access	2 = NACK	-1	q(because it has to send the same packet again	
2 = NACK	0	0 = no access	Non-existent cases because in case of NACK, the length of queue will never be zero			
2 = NACK	0	1 = access				
2 = NACK	Greater than or equal to 1	0 = no access	1 = ACK	1	q = q-1	
2 = NACK	Greater than or equal to 1	1 = access	2 = NACK	-1	q(because it has to send the same packet again	

3.1.5 Transition states

Table 3.1 shows the transitions probabilities for each time step. Each state $s(t)$, which depends on PU's queue, is received by the SU. The SU replies in the next time slot with the possible action. Then environment replies with the next state $s(t+1)$ and a reward. A reward or penalty is given for the SU if it succeeded or failed at each time step. Also, the queue is updated, depending on the arrival rate.

For Example, If the queue is empty, a No-FB state will be sent, and the reward is given if the SU accessed. However, there will be a penalty of negative one if it did not access, and the next state will be No-FB for both No-FB cases. On the other hand, if the queue is not empty, and the SU accessed, it gets a penalty, and the next state is NACK because a collision will arise between the PU and the SU since the PU has priority to transmit as soon as a packet is received. However, if the SU did not access, it gets a positive reward as it saved a collision incident, and the next state is ACK, as the PU succeeded in sending its packet. Also, the queue will be updated. When an ACK is received, and the queue is empty, a reward is gained if the SU accessed. However, if it did not access, it gets a negative reward, and the next state is No-FB as well, and the next state for both ACK cases depends on the arrival rates of packets in the queue, which the SU does not know. In contrast, if the queue was not empty, the SU gets a positive reward if it did not access, then the queue decreases by one, and the next state is ACK as the PU succeeds in its transmission. However, if it accessed, it gets a penalty, and the queue remains the same as the collided packet needs to be retransmitted, and the next state is NACK. The NACK state is an indication for the SU to back off. When a NACK is received, the SU gets a positive reward if it backs off, and the next state will be ACK, and the queue decrease by one. Nevertheless, if it accessed, it gets a penalty, and the next state is NACK, and the queue is the same.

After introducing our POMDP model and the MAC design policy, it should be noted that in a real-life scenario we might not be able to construct our belief vectors $\mathbf{b}$ based on the transition probability function $T(.)$. For example, if the SU does not know λ_p then it will not be able to construct the belief vector, $\mathbf{b}$, of the PU. Therefore, and unlike the work in [21], we propose to implement an RL based MAC that can efficiently learn in a model-free systems in which the underlying dynamics are not fully characterized. Next, we will present different RL algorithms that can be employed to design our SU MAC scheme and compare their performance later via extensive simulations.

3.1.6 Epsilon-Greedy Q-learning algorithm

Q-Learning is one of the most common approaches in reinforcement learning. As in [35] and [36], at each time step, the SU in state s_t chooses an action a_t and goes to the next state s_{t+1} while receiving reward r_t. Through computing and collecting the experiences s_t, a_t, r_t, and s_{t+1}, the RL agent (SU in our case) can compute the state-action value function $Q(s, a)$, which is the expected overall future discounted reward when the SU takes an action a in state s. The update equation for the Q-values is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \underset{a}{max}\, Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (3.12)$$

Algorithm 1: The Q-learning Algorithm

```
REPEAT for each arrival rate
  INITIALIZE Q-table of dimensions $(numstates) * (numactions)$ with all zeros
  INITIALIZE environment
  REPEAT for each episode
    INITIALIZE state
    Reset $queue = 0$
    Perform epsilon decay step
    REPEAT for each time step
      Take action a on env based on epsilon-greedy policy
      Update state
      Collect reward
      Update queue
      Update Q-table using the following equation:
      $Q(state, action) \leftarrow Q(state, action) + (learningrate * (reward + discountfactor * (maxQ(nextstate, allactions))))$
  Calculate throughput
```

In the given formula, two major parameters that influence the equation are α and γ. The parameter α is the learning rate where $0 < \alpha \leq 1$. It affects the extent to which the Q-values are changed by following an action. On the other hand, γ denotes the discount factor. It takes values in the range from zero to one and controls the influence of future rewards on the Q-value. The Q-Learning algorithm builds a lookup table that lists a Q-value for each state-action pair. The SU optimal policy is the deterministic policy that selects the action that has the highest Q-value in each state. Therefore, the optimal policy, $\pi^*(s)$, is given by

$$\pi^*(s) = \arg\max_a Q(s, a).$$

To learn the Q-values, we adopt the ϵ-greedy approach [25]. Accordingly, the agent decides between exploiting the best-known action so far (i.e., the one that results in the highest Q-value) and exploring new actions that might result in higher rewards. The parameter ϵ is used as the probability of exploring new actions. Usually, we start with a high ϵ to explore more and not be trapped in a local minimum. The value of ϵ should decay with time.

As shown in Algorithm 1, for each arrival rate, we start our code by initializing the Q-table with dimensions $(num_states) * (num_actions)$ with zeros values. We initialize the environment code by importing the state, the reward, and the queue with initial values of zero. Afterward, for each episode, based on epsilon, which is the exploration term, we choose our action, either to be random or by selecting the maximum action in the Q-table if it was not zeros. The Q-learning equation calculates the maximum action, then restored in the Q-table. By each episode, the epsilon number decreases by a factor to give more space for exploiting the learned information and actions in the Q-table [25, 37, 38, 39].

3.1.7 Deep Q Network (DQN) algorithm

Deep Q-Learning is a combination of Reinforcement Learning and Deep Learning. DQN emerged to solve infinite-state MDP problems by learning a parametric approximation to the $Q(s, a)$ function. Moreover, the ϵ-greedy policy is used with DQN. This gives more opportunity for random exploration of actions as our machine learns to approximate the Q-values. Through exploration, the SU will explore a variety of actions in different states at different arrival rates [40].

By discretizing the action space (i.e., discretizing the values of the access probabilities), DQN can be used to learn the best action for each PU feedback state. We model the access decision as a classification neural network, and for each feedback state, the machine should output the best action from the set of discrete actions. The machine does this by estimating the Q-value for each action (i.e., the machine will have several outputs that equal the number of actions, each estimating the Q-value corresponding to a specific action). The best action at each state will be the action that results in the highest Q-value [40].

For forward and backward propagation, the expected Q-value for the state is called the target function. The Q-value related to the action taken by the agent is updated while the other actions' Q-values are kept untouched. The target function uses the Bellman equation to estimate the value function of the action as in equation (3.12). Therefore, we have the following update rule for the target Q-value at the $i+1$-th iteration

$$Q^{i+1}(s,a) = r + \gamma \underset{a'}{max}\, Q^{i}(s',a'), \tag{3.13}$$

where s represents the state, a corresponds to the action, r is the instantaneous reward, and s' is the next state. Equation 3.13 estimates $Q(s,a)$ as the reward plus the maximum predicted Q-value of the following state discounted by γ, which is a discount factor as defined before. Eventually, this value is supposed to be predicted and learned by the model for every pair (s,a). After estimating the action-value target function, the best policy is shown to be a greedy policy, i.e., in each state s, the policy selects the action a as $\pi^{*}(s) = \arg\max_a Q^{*}(s,a)$ [40].

The loss function we adopt in our work is the Mean square error (MSE), which calculates the square of the difference between the predicted Q-value and the current estimate of the Q-value as follows [40]

$$L(\theta^{(i)}) = E\left[(y^{(i)} - Q(s,a;\theta^{(i)}))^2\right]$$

$$y^{(i)} = E\left[r + \gamma \underset{a'}{max}\, Q(s',a' : \theta^{(i-1)})|s,a\right],$$

where $\theta^{(i)}$ represents the set of the neural network parameters at iteration i.

As shown in Algorithm 2, for each arrival rate, we start by initializing the Q-table with zero weights. The input is the states, and the output is the values of each action. We initialize the algorithm by importing the state, the reward, and the queue with initial values of zero. Afterward, based on epsilon, the action is selected to be random for exploring or by selecting the action with the highest Q-value. Also, the next Q-value for the next state and action is calculated. Then, we subtract both the current predicted Q-value with the maximum possible Q-value for the next state. By calculating the error difference, we keep doing the gradient for the equation until reaching zero difference, and hence, convergence. By each episode,

Algorithm 2: The Q-learning algorithm

```
REPEAT for each arrival rate
  INITIALIZE Neural Network with zero weights. The
  input and output dimensions of the neural network are
  num_states and num_actions, respectively.
  INITIALIZE environment
  REPEAT for each episode
     INITIALIZE state
     Reset $queue = 0$
     Perform epsilon decay step
     REPEAT for each time step
        Take action a on env based on epsilon-greedy policy
        Update state
        Collect reward
        Update queue
        Update the weights using the following equation:
        $Loss = meansquareerror[reward + discountfactor * (max(predicted$
        $valuesofQ(nextstate, allactions))) - predictedvalueofQ(state,$
        $action)]$
  Calculate throughput
```

the epsilon number decrease by a factor to give more space for exploiting the learned information and actions in the Q-table [40, 41, 42].

3.1.8 Deep Deterministic Policy Gradient (DDPG)

DDPG is an off-policy model-free algorithm. It is a combination of policy gradient and DQN in order to learn continuous action space. It uses experience replay and target networks to stabilize the training process. Experience replay, which store (reward, action, $S(t)$, $S(t+1)$), is used to break temporal correlation between inputs. on the other hand, target networks are used to converge faster by updating the networks slowly to keep the estimated targets stable and separable from the actual networks. In DDPG architecture, we have four neural networks: two for the actor $\mu(s|\theta^{\mu})$ and the critic $Q(s, a|\theta^{Q})$ as shown in Fig. 3.3 and the other two for the target actor $\mu'(s|\theta^{\mu'})$ and the target critic $Q'(s, a|\theta^{Q'})$[26, 27, 34]:

The steps of the DDPG algorithm can be summarized as follows:

1. An actor-network, critic network, target actor, target critic network, and replay buffer are initialized.

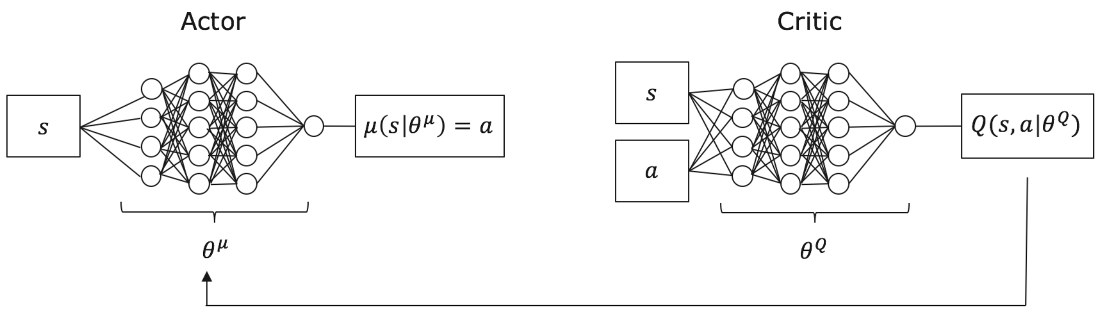

FIGURE 3.3: Deep Deterministic Policy Gradient

2. The state is passed to the actor to estimate the deterministic policy. For better exploration, an Ornstein–Uhlenbeck (OU) noise is added to the action.

3. The deterministic policy and the state are passed to the critic network to produce the $Q(s, a|\theta^Q)$. It learns to estimate the best action value (Q-value) based on the best action from the Actor-network.

4. The target networks are frozen and are updated after every few episodes for better convergence stability.

5. Each time step, the agent takes action A and receives reward R and moves to new state

6. The agent's state, action, reward and next state are stored in the replay buffer in order to training take samples that are not correlated.

7. After training the networks using a mini batch of samples, the critic loss is calculated as the Mean-Squared Error (MSE) between the $Q(s, a|\theta^Q)$ and $Q'(s, a|\theta^{Q'})$ as in equation (2.14) and (2.15) like in DQN but here, the $Q'(s, a|\theta^{Q'})$ is taken from the target critic network.

8. To minimize the loss, a gradient descent is applied .

9. For the actor network, gradient ascent is applied as in equation (3.14)

$$\nabla_{\theta^{\mu}}\mu = E_{\mu}[\nabla_{\theta^{\mu}}Q(s, \mu(s|\theta^{\mu})|\theta^{Q})] = E_{\mu}[\nabla_{a}Q(s, a|\theta^{Q}) \cdot \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})] \tag{3.14}$$

10. Repeat steps until convergence or reaching maximum number of iterations.

3.1.9 Simulation models

We compare our result with two conventional methods which are baseline system 1, and baseline system 2.

- Baseline system 1: The feedback (FB) system'SU relies on accessing the channel if there is an ACK or nothing is received. The optimum access probability is calculated by solving the balance equation based on MC model as shown in equation (3.15) [19].

$$a_s^* = min(1 - \lambda_p / 2\lambda_p, 1) \tag{3.15}$$

- Baseline system 2: The greedy algorithm is based on calculating the belief probability based on the history of the PU queue being empty or not, then comparing it to a threshold and take access decision for the SU if higher, or refrain from accessing if lower as shown in equation (3.16).

$$GreedyAlgorithm = \begin{pmatrix} access & if & b(Q_t = 0_F) > w/(1+w) \\ no-access & if & b(Q_t = 0_F) \leq w/(1+w) \end{pmatrix} \tag{3.16}$$

  In the greedy approach, if the SU accesses in case the PU's queue was empty, it will take a positive reward. However, if it didn't access, it will not be given anything. In case it accessed simultaneously with the PU, it will take a negative reward $-w$ as shown in equation (3.17). In the simulation, $w = 0$ means that the SU will always attempt transmission in the ACK or No-FB case. As a consequence, it degrades the SU throughput at high PU arrival rates due to frequent collisions. As w increases, there is less tendency for the SU to access the channel. Hence, losing more transmission opportunities over the channel for low PU arrival rates.

$$R(s,a) = \begin{pmatrix} 1 & a = access & s = 0_F \\ 0 & a = no - access & \forall s \\ -w & a = access & s \neq 0_F \end{pmatrix} \tag{3.17}$$

3.1.10 ARQ performance results

In our simulations, for discrete action space case, Q-Learning and DQN, we assume eleven possible actions for the SU: $\mathcal{A}$ = {no access, access with probabilities $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. We start with $\epsilon = 1$ to favor exploration as the SU begins to learn to explore all sates, and actions in all arrival rate . We also set a decaying ϵ; the decaying factor is 0.999, with a min exploration rate of 0.01 to guarantee certain level of exploration. The learning rate is set to be $\alpha = 0.0001$. The discount factor $\gamma = 0.95$, which is the discount factor for future rewards [36, 42, 43].

Based on the throughput metric, several parameters were tuned to achieve maximum SU throughput, and the models were retrained multiple times to achieve maximum throughout in all RL systems.
Through observation, Setting $epsilon = 1$ is to converge faster as we have eleven actions that need to be explored in every arrival rate. Moreover, there are two access decisions inside each of the eleven actions, which are the probability of access or no access. The access decision is based on the probability percentage of the action. So, it takes long to explore the best action all states. For example, the No-FB state at high arrival rate 0.8 and 0.9 is not visited frequently compared with the ACK state in the ARQ feedback system. Hence, it was beneficial to start at $epsilon = 1$ to guarantee that all states are visited. However, acting greedy or exploiting was crucial, so we set a decaying $epsilon = 0.999$ for the SU to start acting greedily after some time of collecting information about the PU activity and knowing its pattern. Also, if we set $epsilon = 1$ for a long time, the PU queue will rapidly increase, and then the SU will decides not to access, especially at the high arrival rates and result in a bad throughput. Lastly, a minimum exploration

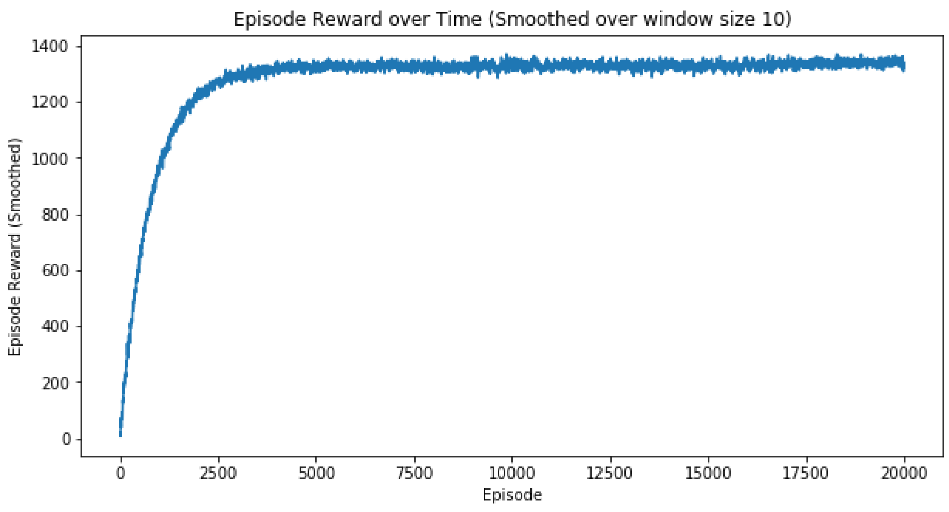

FIGURE 3.4: Rewards vs number of episodes

value is maintained to guarantee a certain level of exploration. Due to the fact that, in the low arrival rate 0.1, the SU refrains from accessing in the NACK state and converges fast, unlike the No-FB in the high arrival rate, which requires exploration.

It is noticed that the RL SU agent accesses with probability 1 most of the time until 0.3 arrival rate without affecting the PU QoS. On the other hand, Starting from 0.7 arrival rate, it backs off. Most of the optimization is done in the middle range arrival rate.

It is observed that setting $\alpha = 0.0001$ stabilizes the action-value function. So each time the code run, the SU takes the same actions, unlike if we increase it to 0.001. It will converge faster, but will give different close actions each time. Thus, it is better to decrease the learning rate and increase the convergence time to reach more stable system.

Fig. 3.4 shows the accumulated rewards for the SU against the number of episodes for ARQ FB-system using Q-learning algorithm with $r = 1$. The SU reached convergence at episode number 5000. Similarly, Fig. 3.5 shows the accumulated rewards for the SU against the number of episodes for the ARQ FB-system using DQN algorithm with $r = 10$. The SU reached convergence at episode number 1000.

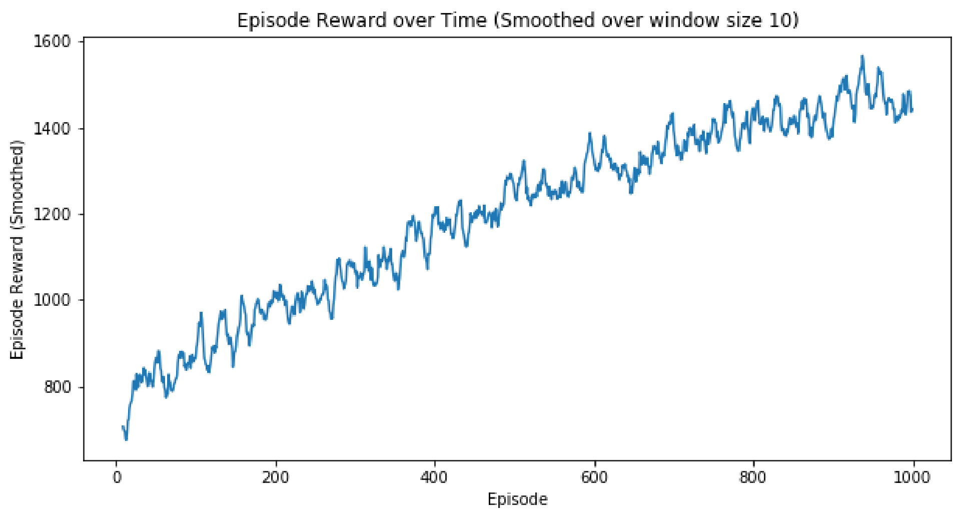

FIGURE 3.5: Rewards vs number of episodes

It is noticed that increasing the SU's reward for its success access encourages the SU to access more in the No-FB state and less in the ACK state. However, pushing the reward for more than 50 in the ARQ system results in a high collision. Thus, high number of packets will be accumulated in the PU queue. As a result, an unstable system will appear with low throughput. On the other hand, if a penalty is applied to the SU in case of collision, it will adopt non-accessing behavior in all arrival rates. Thus, a moderate tuning is preferable to achieve best result.

First, we investigate the effect of the reward function on our SU throughput system performance. We compare five different values for r, as shown in Fig. 3.6 for the Q-Learning based approach. The reward is given for the agent in case it accessed and succeeded when the queue is empty as it may access but make a collision with the PU. As a result, a penalty will be applied. However, higher values for r resulted in slightly degraded performance since they caused the SU to aggressively access the channel, which resulted in collisions with the PU. The algorithm achieved the best performance with $r = 1, 2, 10, 50$ with a slight difference in the resulting access probabilities (actions) for different arrival rates. The best performance corresponds to the case of RL with $r = 50$. It causes the SU to greedy access the channel compared to the cases with a lower r. For example, at 0.5 arrival rate, the access probability is 1 for the No-FB and 0.1 for ACK state.

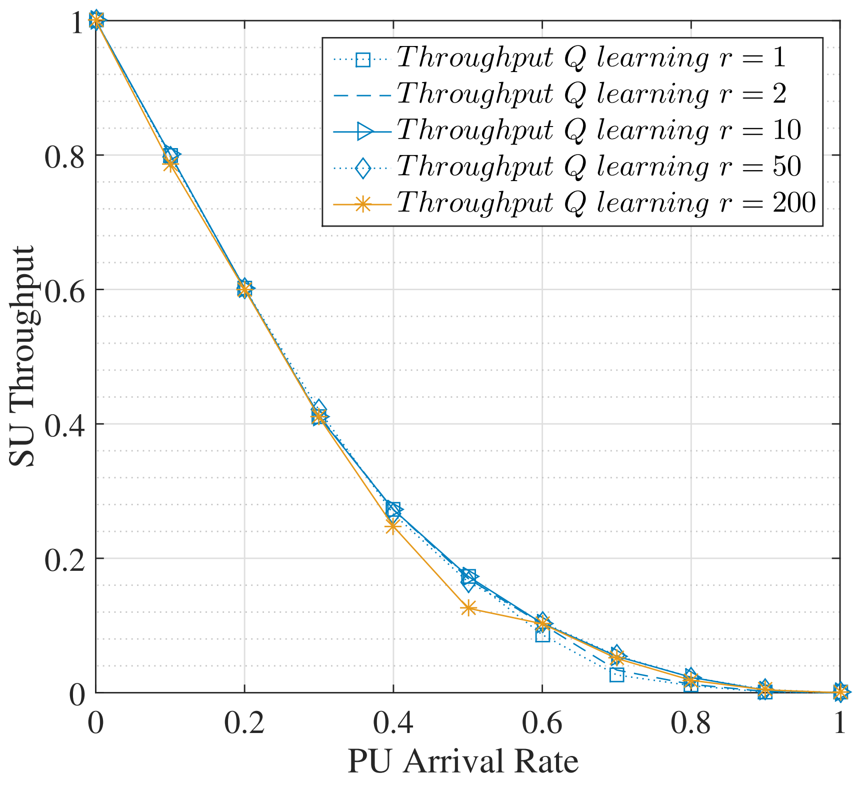

FIGURE 3.6: ARQ feedback using RL (Q-Learning) with different rewards

However, with $r = 200$, the access probabilities are 1 for the No-FB and 0.9 for the ACK state, which caused collision and slight degradation.

Fig. 3.7 depicts a performance comparison in terms of the SU throughput between the Q-Learning $r = 50$ algorithm, the greedy algorithm proposed in [21], and the FB-based approach proposed in [19]. The best performance relates to Q-Learning algorithm. Although the conventional methods assumes the knowledge of the arrival rate, they were accessing only in the ACK or No-FB state. However, our proposed SU model access in the No-FB and ACK states, which showed better performance. Moreover, our model works in an incomplete environment.

Fig. 3.8 shows a comparison between two ARQ Q-learning algorithms. One used the last PU feedback, while the other used the last two. Starting from 0.3 to 0.6 arrival rate , the results showed that the two feedback systems slightly outperformed the one feedback system, because the SU succeeded more in estimating the behavior of the PU in the next time slot.

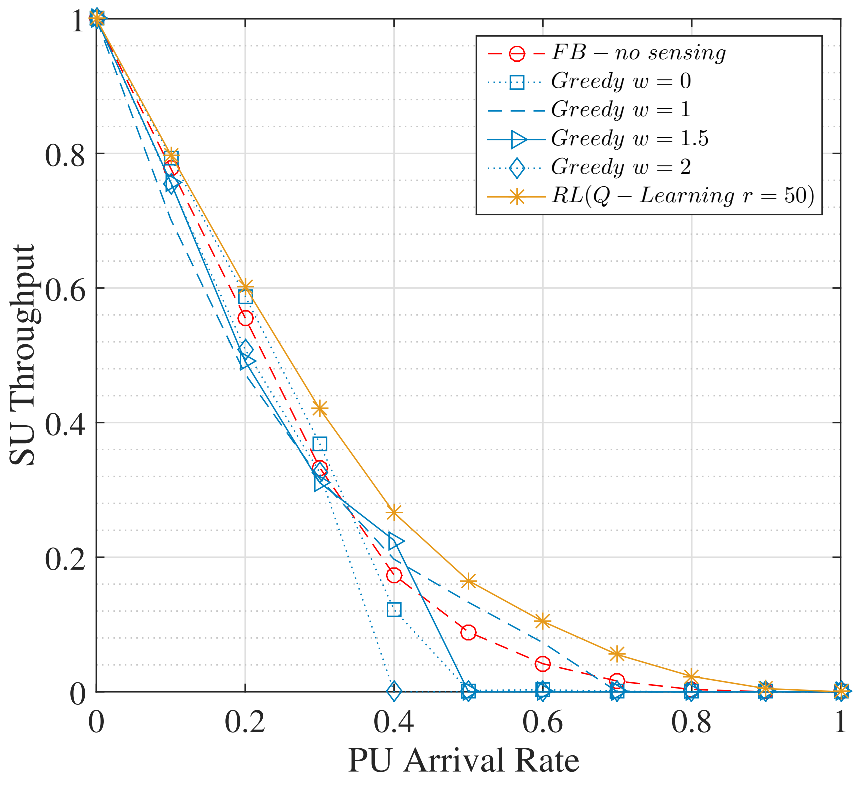

FIGURE 3.7: ARQ feedback-based access using RL (Q-Learning) with $r = 50$ versus the greedy algorithm [21] and the FB-based approach of [19]

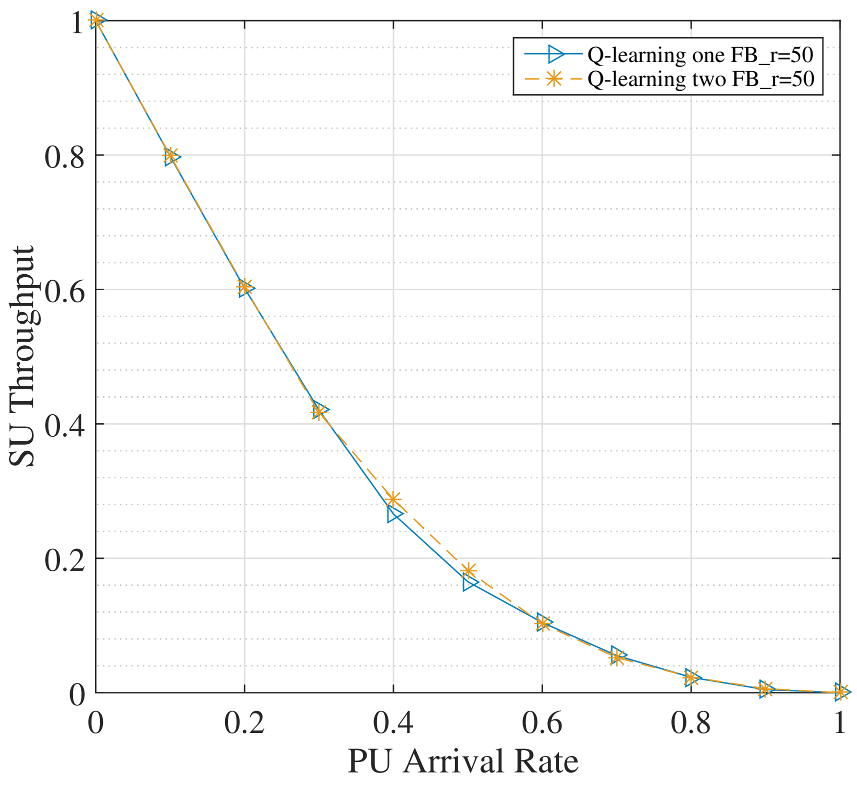

FIGURE 3.8: ARQ using the last FeedBack VS the last Two FeedBacks

The slight improvement is only achieved in the middle range due to the fact that before 0.3 arrival rate both systems access with probability one in the No-FB and the ACK. On the other hand, after 0.6 arrival rate, the SU starts to refrain from accessing in both system showing similar performance. So, the only range of improvement is the middle range. Thus, the more history or information we know about the PU, the more estimation for its behavior, which results in higher throughput.

Next, we consider the DQN approach. The deep Q-Learning network architecture consists of three layers: the input layer, one hidden layer, and the output layer. First, the network input layer is composed of three nodes of one-hot encoded state vector (3×1). It represents the three observations (ACK, NACK, and No Feedback). Second, we have a hidden layer with ten fully connected neurons (10×1) with a Sigmoid activation function. Finally, the output layer consists of eleven nodes that correspond to the eleven actions of the SU (11×1) with a linear activation function. The optimizer used

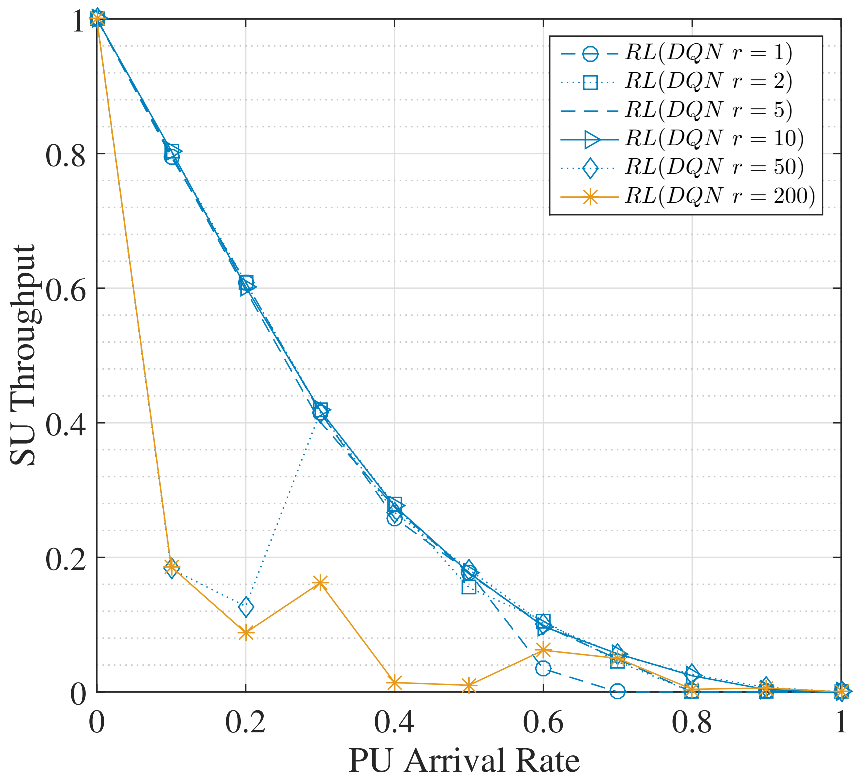

FIGURE 3.9: ARQ feedback using RL (DQN) with different rewards

in the prediction is the Adaptive Moment Estimation (ADAM) [41].

Fig. 3.9 shows the effect of the reward function value on the performance of the DQN-based approach. Similar to Fig. 3.6 in Q-Learning algorithms, again, the value of r presents a trade-off between aggressively accessing the channel for higher r values and missing transmission opportunities for smaller r values. All algorithms show same performance at the low arrival rate. However, at high arrival rate, the best performance is achieved by $r = 10$ algorithm as $r = 2$ algorithm showed slight degradation at 0.8 arrival rate. Also, $r = 1$ algorithm showed less performance at 0.6 arrival rate, and refrained from accessing at 0.7 arrival rate. $r = 50$ algorithm showed waek performance at 0.2 arrival rate due to its aggressiveness to access the channel. $r = 200$ algorithm showed the worst degradation case as a result from the collision between the PU and the SU.

Similar to the Q-Learning, Fig. 3.10 compares the performance of the

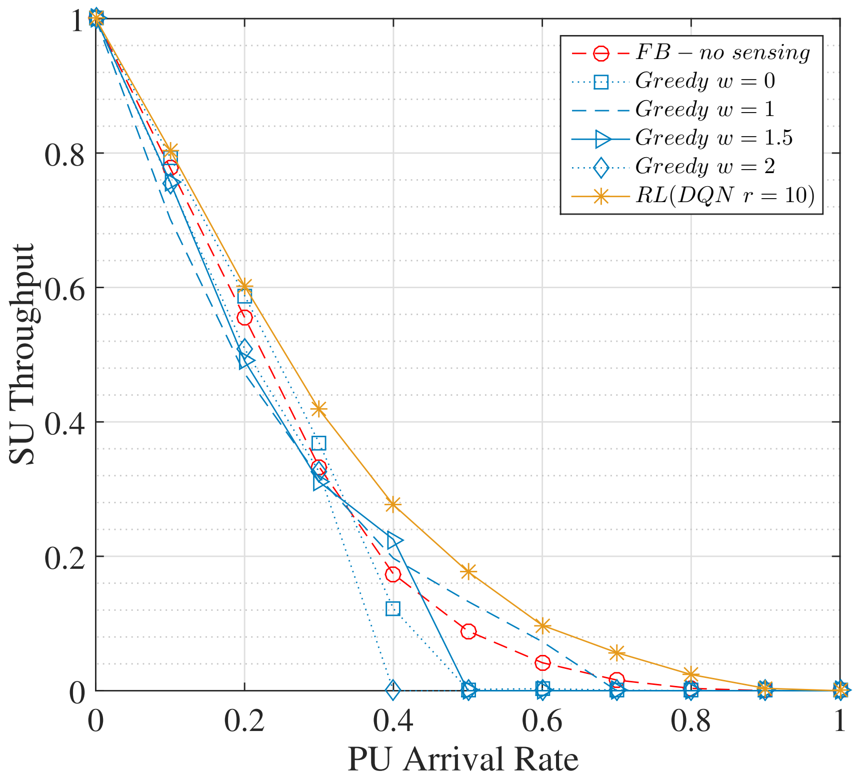

FIGURE 3.10: ARQ feedback using RL (DQN with $r = 10$) vs conventional methods

DQN approach for $r = 10$, which showed the best performance, to the greedy algorithm for different w values in [21] and also to the FB-based approach in [19]. The DQN-based approach outperforms the two access schemes while assuming minimal information about the primary user (e.g., it does not assume any knowledge of the PU arrival rate while the other approaches assume perfect knowledge of the PU arrival rate).

Comparing RL algorithms, we compare Q-Learning $r = 50$ to DQN $r = 10$ in Fig. 3.11. The two algorithms show similar SU throughput with close access probabilities. Both algorithms use discrete action space, and value-based function dependant. In Q-learning, the agent needs to visit each state-action pair finite number of times to get the true Q-values and converges. On the other hand, DQN is a function-approximator that learns mapping between the state-action pairs and their Q-values in case of large

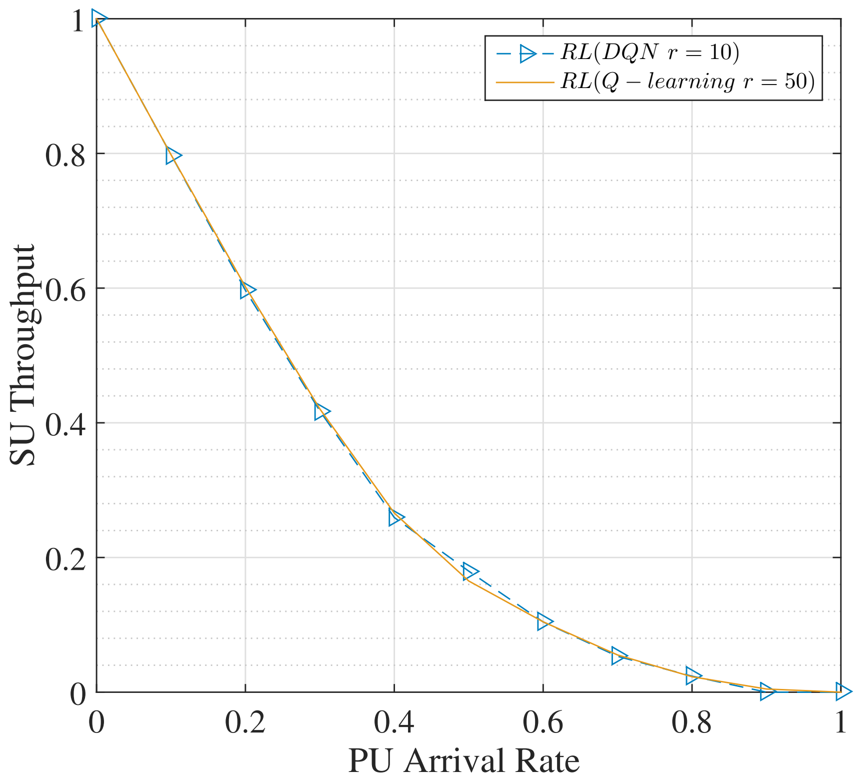

FIGURE 3.11: ARQ feedback using RL (DQN vs Q-Learning algorithm)

input state space that can not be visited individually. DQN is better in feature extraction and taking the right action due to its neural network's feature extraction. However, in our ARQ case, the input state is only three bits of one hot encoder, which does not require high dimensional feature extraction tool and can be easily handled and converged by Q-learning. As a result, both algorithms showed similar performance [40, 44, 45].

Next, we consider the DDPG-based approach. The actor-critic network architecture comprises four layers: one input layer, two hidden fully connected layers of 64 neurons, and one output layer. ADAM is used to learn the neural network parameters with a learning rate of 10^{-4} and 10^{-3}, respectively. A non-linear rectifier is used in all hidden layers beside a tanh function in the output layer for bounding the actions. The mini-batch size is set to be 64, and the replay buffer size is set to be 10^{6}. For exploration, Ornstein-Uhlenbeck noise was added to the action output [46, 47].

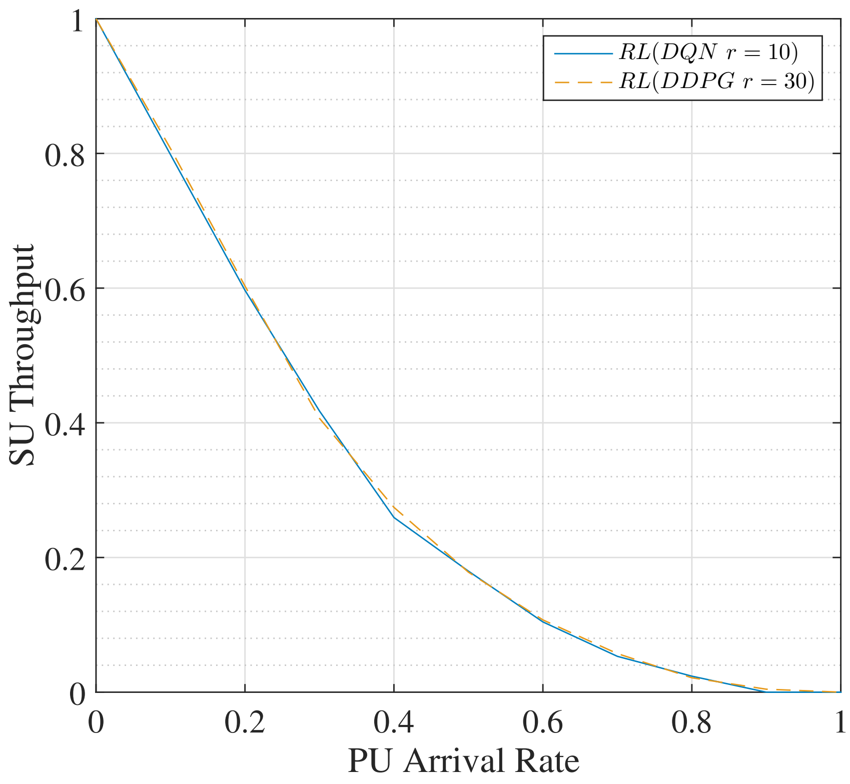

FIGURE 3.12: ARQ feedback using RL (DQN vs DDPG)

We compare DQN to DDPG in Fig. 3.12, which shows comparable performance for both algorithms. However, DDPG has minor gain over DQN. This is because DDPG has a continuous action space, while DQN is restricted to discrete action space. For example, at 0.6 arrival rate, DDPG access with probability of 0.17867616 in the ACK and 1 in the No-FB, which results in a throughput of 0.10732, whereas DQN accesses with probability 0.2 in the ACK and 1 in the No-FB, which results in a slightly lower throughput of 0.104. Finally, the continuous access probability of DDPG led to a minor decimal throughput gain. This minor access probabilities and throughput gain, could have a huge impact on/or result in a successful access attempt, or saving call drop, which is crucial in some applications as military, business, and emergency cases [46, 48, 49].

$$a_s^* = (1 - \lambda_p) \setminus 2\lambda_p, \tag{3.18}$$

Fig. 3.13 shows the SU access probabilities in the ARQ feedback scheme

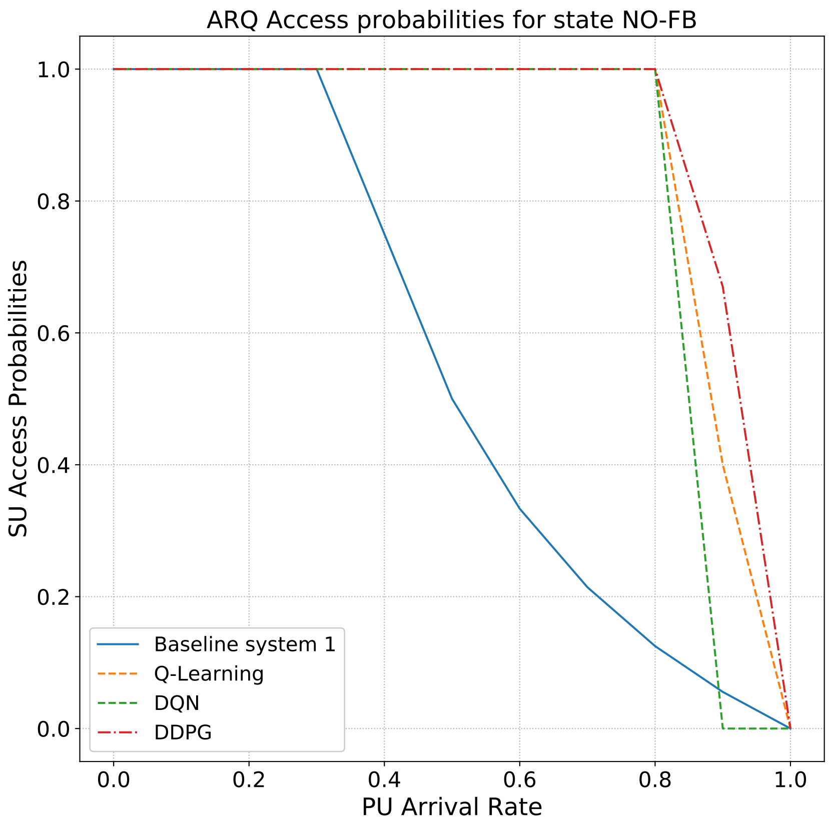

FIGURE 3.13: ARQ Access probabilities for No-FB state

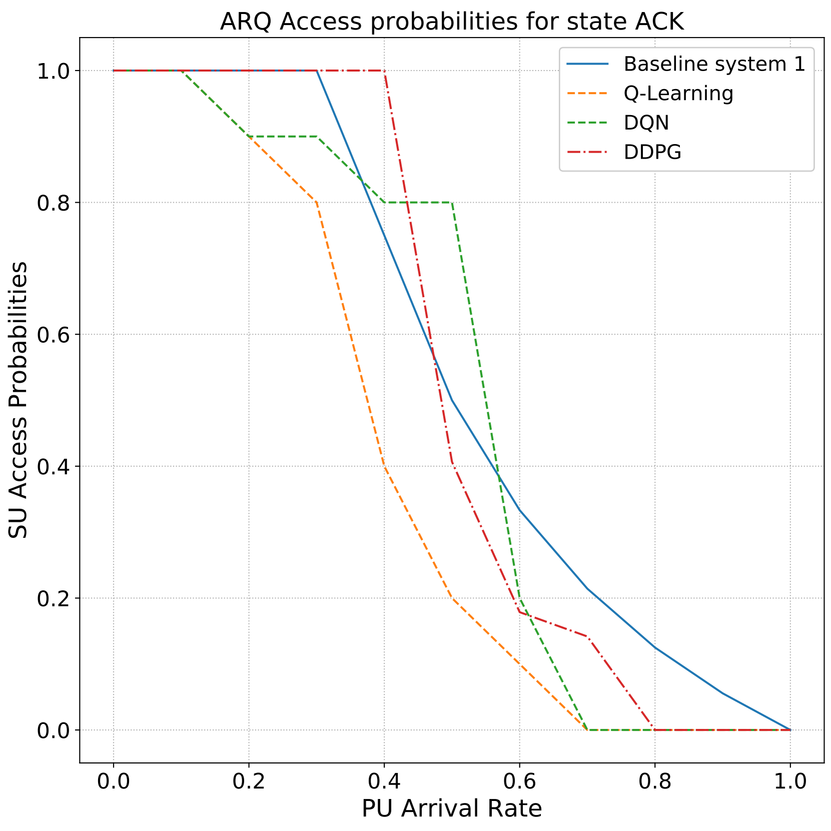

FIGURE 3.14: ARQ Access probabilities for the ACK state

at the (No-FB state). It compares the old method (FB-based system) with Q-learning, DQN, and DDPG algorithms. The old method relied on a closed-form expression of equation (3.18) from [19] for the access probabilities, assuming it knows the arrival rate. Q-learning, DQN, and DDPG access with probability one until 0.8 arrival rate, because of the high reward given and greediness to access. The rewards for accessing for the three algorithms are 50, 10, and 30 respectively. However, that did not affect the PU QoS. The access with probability 1 until 0.8 arrival rate. Starting from 0.8 arrival rate, all RL algorithms showed a sharp decline in the access probabilities to avoid collision with the PU. DQN shows the sharpest decline, while, DDPG shows smoother decline because of its continuous action space.

Fig. 3.14 shows the SU access probabilities in the ARQ feedback scheme at the (ACK state). It compares the old method (FB-based system) with

Q-learning, DQN, and DDPG algorithms. The SU agent tends to be less aggressive toward accessing the ACK state than in the No-FB state in all algorithms. This is due to the RL agent's (SU) sensitivity to guess the probability of PU arrival in the next time slot of the ACK state than the No-FB. The probability of arrival in the ACK is more than the No-FB state. DDPG algorithm showed slightly better performance. For example, at 0.4 arrival rate, Q-learning, DQN, and DDPG access with probability 0.4, 0.8, and 1. As a result, DDPG showed minor throughput gain, because it accesses with high probability in the No-FB and the ACK states, as shown in Fig. 3.12 .

3.2 Channel Quality Indicator (CQI) based access scheme

In this section, the system is using HD sensing with the CQI feedback values. The SU is assumed to have access to the PU CQI feedback. We assume that we have a binary CQI feedback in the form of "GOOD" or "BAD" channel. The PU transmits its packet as soon as arrival happens in its queue in the good CQI state. Upon receiving (good) channel CQI feedback, the SU transmits and access the PU channel with an access probability of as, after HD sensing, and based on the arrival rate. on the other hand, Upon receiving (bad) channel CQI feedback, the SU access the PU channel with probability one as the PU refrains from accessing the channel to save failed transmission [19].

3.2.1 Channel model

The PU channel is modeled as "GOOD" and "BAD" states. Let p_G denotes the probability of the channel being in the good state, and p_B denotes the probability of the channel being in the bad state, as shown in Fig. 3.15. Let ζ_G and ζ_B be the steady-state probabilities of the channel being in the good and bad states, respectively, as shown in equation (3.19) [19].

$$\zeta_g = \frac{1-p_B}{2-p_B-p_g}, \quad \text{and} \quad \zeta_B = \frac{1-p_g}{2-p_B-p_g} \tag{3.19}$$

Further, it is assumed that the channel state does not change during a one-time slot. A collision channel model is assumed, i.e., if both the PU and SU transmit their packets simultaneously in the next time slot, then the packets collide with each other, and both packets are lost.

The closed form expression of the SU throughput for the perfect sensing curve can be shown in the following equation (3.20) [19]:

$$\mu_{sp} = \begin{cases} 1-\lambda_p, & if\ \lambda_p \geq \zeta_B - 1. \\ \zeta_B, & \text{otherwise} \end{cases}. \tag{3.20}$$

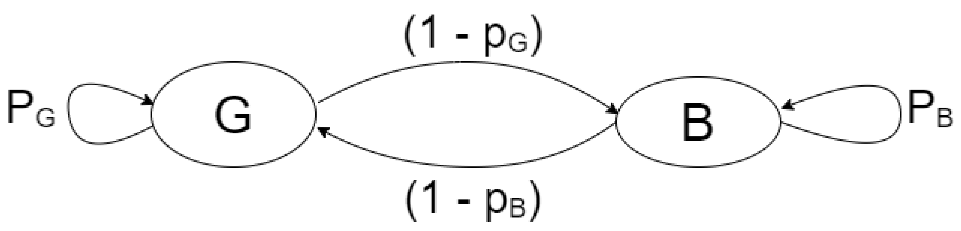

FIGURE 3.15: The channel model

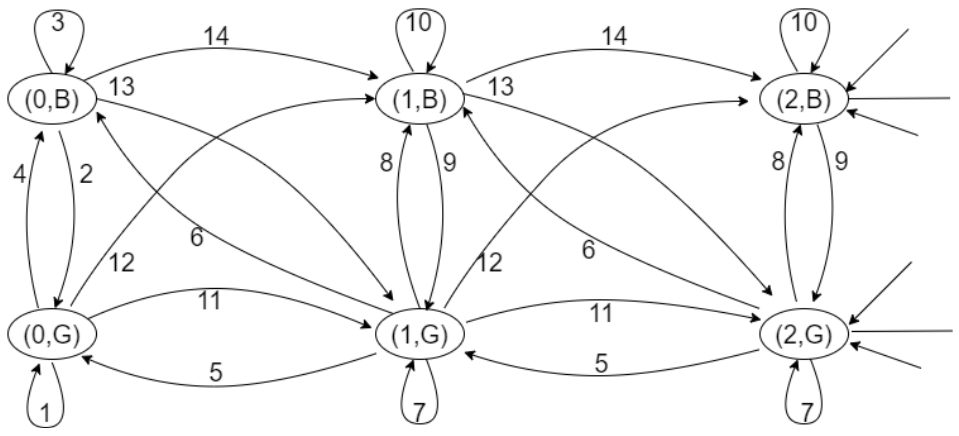

FIGURE 3.16: Two-dimensional MC model for the PU queue

3.2.2 System model

We consider a CR system that has a system model, as shown in subsection 3.1.1.

3.2.3 PU queue model

We present our PU queue as a two dimensional Markov chain model as shown in Fig. 3.16. It consists of two types of states (K, G) and (K, B). K denotes the number of packets in the PU's queue. Whereas, G and B denote that the PU's channel is in the good or bad state. This strategy models the SU access decision scheme as a POMDP.

The transitions between states are as follows:

From (K, G) to $(K+1, G)$: in this case, the transition occurs according to the following equation: $Pr(X(n+1) = (K+1, G)|X(n) = (K, G))$ = Pr((a new packet arrives at the PU queue) ∩ (SU does not detect the PU presence and decides to access the channel) ∩ (the channel in the next time slot remains in the good state))= $\lambda_p as(1-p_d)pg$, where p_d is the detection probability of

TABLE 3.2: Transition probabilities of Markov chain of the CQI feedback- (HD) system.

Number	Transition Probability
1	$((1-\lambda_p)+\lambda_p((1-a_s)(1-p_d)+p_d))p_G$
2	$(1-\lambda_p)(1-p_B)$
3	$(1-\lambda_p)p_B$
4	$((1-\lambda_p)+\lambda_p((1-a_s)(1-p_d)+p_d))(1-p_G)$
5	$(1-\lambda_p)((1-a_s)(1-p_d)+p_d))p_G$
6	$(1-\lambda_p)((1-a_s)(1-p_d)+p_d)(1-p_G)$
7	$(\lambda_p((1-a_s)(1-p_d)+p_d)+\lambda_p a_s(1-p_d))p_G$
8	$(\lambda_p(1-a_s)+(1-\lambda_p)a_s)(1-p_G)$
9	$(1-\lambda_p)(1-p_B)$
10	$(1-\lambda_p)p_B$
11	$\lambda_p a_s(1-p_d)p_G$
12	$\lambda_p a_s(1-p_d)(1-p_G)$
13	$\lambda_p(1-p_B)$
14	$\lambda_p p_B$

the spectrum sensor.

From (K,G) to $(K+1,B)$: it is same as the above transition but here the term p_g is replaced by $(1-p_g)$, which indicates the probability of the next time slot is bad. Therefore, the transition probability is equal to: $\lambda_p as(1-p_d)(1-p_g)$.

The rest of the transition probabilities can be deduced easily from Fig. 3.16 and listed in Table 3.2 [19].

3.2.4 POMDP environment framework

A POMDP is defined by the tuples $(\mathcal{S},\ \mathcal{A},\ \mathcal{O},\ T,\ \Omega,\ R)$, where the set $\mathcal{A}$ denotes the set of SU actions (which correspond to different SU access probabilities). The set $\mathcal{S}$ denotes the PU Markov chain states $S=\{\{i_F\},\{j_R\}\}$, $i=0,1,\cdots$ and $j=1,2,\cdots$. The set $\mathcal{O}$ defines the observations set that is presented by $\mathcal{O}=\{\text{Good},\ \text{Bad},\}$ [19].

The function $T(.)$ denotes the transition probabilities function, where $T(s'|s,a)$ indicates the likelihood to go from state s to state s' given action

a. To better illustrate it, and following the formulation in [21], which is reproduced here for convenience, we give few examples below for $T(s'|s,a)$, for different values of s, s' and a [19].

T((i,B) | (j,G), no access) = 0, $\forall j \leq i$

$$T((i,B)|(j,G),\ \text{no access}) = 0,\ \forall j \leq i$$

$$T((i,B)|(j,G),\ \text{access}) = 0,\ \forall j > i$$

$$T((i,G)|(j,G),\ \text{access}) \quad = 0,\ \forall i,\ j \neq 0$$

$$T((1,G)|(0,G),\ \text{access}) \quad = \lambda_p,$$

$$T((1,G)|(0,G),\ \text{no access}) \quad = \lambda_p,$$

$$T((i,G)|(j,B),\ \text{no access}) = 1-\lambda_p,\ \forall i = j, j \neq 0$$

$$T((i,B)|(j,G),\ \text{access}) \quad = \begin{cases} 1-\lambda_p & \text{if } i = j, j \neq 0 \\ \lambda_p & \text{if } i = j+1, j \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

(3.21)

The function $\Omega(o|s', a)$ denotes the probability of observing o given that action a is applied to result in state s'. It can be estimated as [3]

$$\Omega(o|s' = (i, G),\ \text{no access}) \\ = \begin{cases} 0 & o = \text{Bad and } \forall s' \\ P_{\text{G}}(s') & o = \text{Good, } s' = (0, G) \\ P_{\text{B}}(s') = 1 - P_{\text{G}}\ (s') & o = \text{Bad, } s' = (0, G) \\ P_{\text{G}}(s') & o = \text{Good, } s' = (1, G) \\ P_{\text{B}}(s') = 1 - P_{\text{G}}(s') & o = \text{Bad, } s' = (1, G) \\ 1 & o = \text{Good, } s' \geq (2, G) \\ 0 & o = \text{No-FB, } s' \geq (2, G) \end{cases} . \tag{3.22}$$

If we begin with a certain vector of belief $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the index of time, then the new vector of belief is given after the action $a_(t)$ observing some $o_(t+1)$. However, the formula values will not be affected since, the reward of SU will always be zero in the no-access state (as described later) [19].

$$\Omega(o|(i, G),\ \text{access}) = \begin{cases} 0 & \forall o \text{ and } i \geq 2 \\ 1 & o = \text{Good, } i = 0, 1 \\ 0 & o = \text{Bad, } i = 0, 1 \end{cases} \tag{3.23}$$

$$\Omega(o|(i, B),\ \text{no access}) = 0\ \forall o \tag{3.24}$$

[3]By abuse of notations, we set $\Pr(A|B) = 0$ if $\Pr(B) = 0$ (for example we set $\Omega(o|i_R,\ \text{no access}) = 0$ since $\Pr(i_R,\ \text{no access}) = 0$ under our collision system model assumption).

$$\Omega(o|(i,B),\ \text{access}) = \begin{cases} 1 & o = \text{Bad} \\ 0 & o = \text{Good} \end{cases}. \tag{3.25}$$

The R, reward function is calculated as follows.

$$R(s,a) = \begin{cases} w & a = \text{access},\ s = (0,G) \\ -1 & a = \text{no access},\ \forall s = (O,G) \\ 1 & a = \text{no access},\ \forall s \neq (O,G) \\ -1 & a = \text{access},\ s \neq 0_F \end{cases}. \tag{3.26}$$

It is an immediate reward that the SU has earned to take a specific action and reach a new state. If the queue of the PU is empty, i.e., $s = 0_F$ and the SU accessed the channel, it will gain a positive reward. However, in this same case and if the SU does not access the channel, it receives a penalty due to the lost transmission opportunity. Moreover, if the queue is not empty and the SU accessed, it also receives a penalty. On the other hand, if the SU does not access the channel, it receives a positive reward for avoiding a certain collision with the PU [19].

The belief vector is given by $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the time index. After taking an action a_t and observing some o_{t+1}, the new belief for some state s_{t+1} at time $(t+1)$ is given by

$$b(s_{t+1}) = \eta\Omega(o_{t+1}|s_{t+1},a_t)\sum_{s_t\in\mathcal{S}} T(s_{t+1}|s_t,a_t)b(s_t), \tag{3.27}$$

As η is considered to be the variable of normalization shown by

$$\eta = \frac{1}{\sum_{s_{t+1}\in\mathcal{S}}\Omega(o_{t+1}|s_{t+1},a_t)\sum_{s_t\in\mathcal{S}}T(s_{t+1}|s_t,a_t)b(s_t)}.$$

[21]

3.2.5 POMDP MAC Policy

We consider a POMDP MAC policy similar, as shown in subsection 3.1.4.

3.2.6 RL algorithms

We propose two RL algorithms to solve the CQI POMDP problem, Q-Learning as shown in subsection 3.1.6 and DDPG in subsection 3.1.8.

3.2.7 Simulation models

We compare our result with the conventional method which is the baseline approach.

- Baseline system: It drives a closed form expression based on a steady state distribution calculation of two dimensional MC model, given the know the arrival rate. The optimum access probability is shown in equation (3.28). The closed form expression for the throughput is shown in equation (3.29). Also, the perfect sensing curve is shown in equation (3.30) where the SU access whenever the PU's queue is empty and in the bad channel state [50].

$$a_s^* = (P_B + \sqrt{(\lambda_p(P_B - 1)(P_B + P_G - 2)} - 1)) \setminus (P_D + P_B - P_D P_B - 1), \tag{3.28}$$

$$\mu_{sp} = \begin{Bmatrix} 1 - \lambda_p & if & \lambda_p < \zeta_B - 1 \\ \zeta_B & otherwise. & \end{Bmatrix} \tag{3.29}$$

$$\mu_s = \zeta_B + \zeta_G(1 - P_f), \tag{3.30}$$

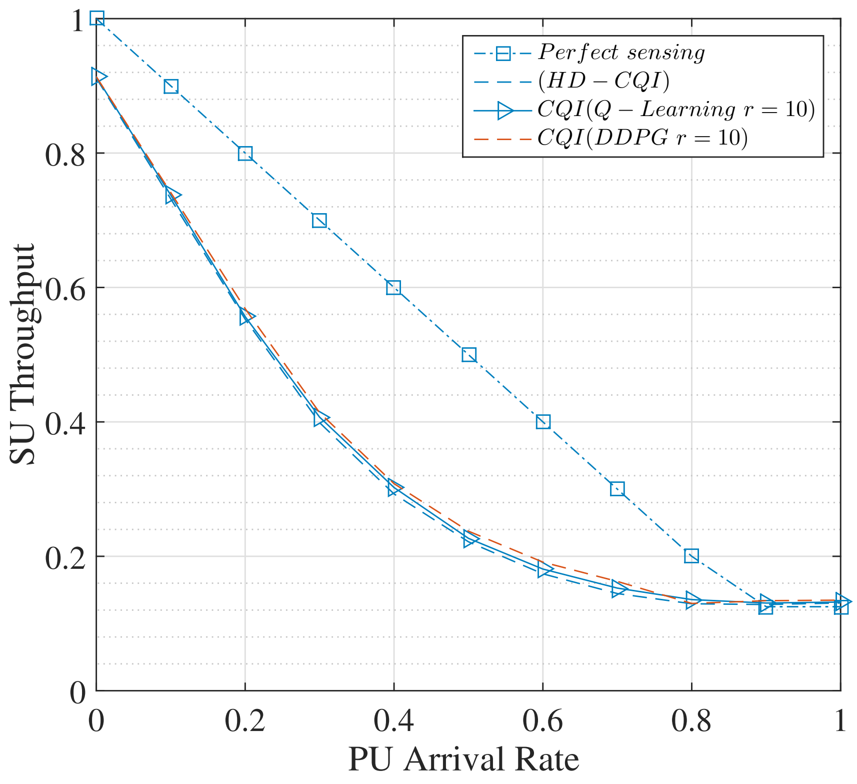

FIGURE 3.17: CQI feedback with RL (Q-Learning, DDPG) vs HD-CQI [8]

3.2.8 CQI performance results

Fig. 3.17 compares the throughput of the three models. The Two proposed algorithms, namely, Q-Learning and DDPG, and the baseline approach of [50], where a closed-form expression is derived. The perfect sensing upper bound curve is drawn based on the equation (3.30). The probability of false alarm p_F is set to 0.1 and the probability of detection p_D is set to 0.9. Moreover, the probability of the channel staying in the good p_G, and the bad are 0.9 and 0.3, respectively. DDPG approach yields the best performance with a minor gain over Q-Learning approach. It is attributed to the fact that DDPG exploits the whole space of action probabilities, and it does not discretize the action space. For example, at 0.6 arrival rate, the SU throughput by Q-Learning in the good state is 0.18104 as the access probability 0.4, while DDPG is 0.19138 as the access probability is 0.41840127, which is slightly better. That was due to allowing continuous action space in the case of DDPG can, in general, result in higher rewards. It should be noted that at

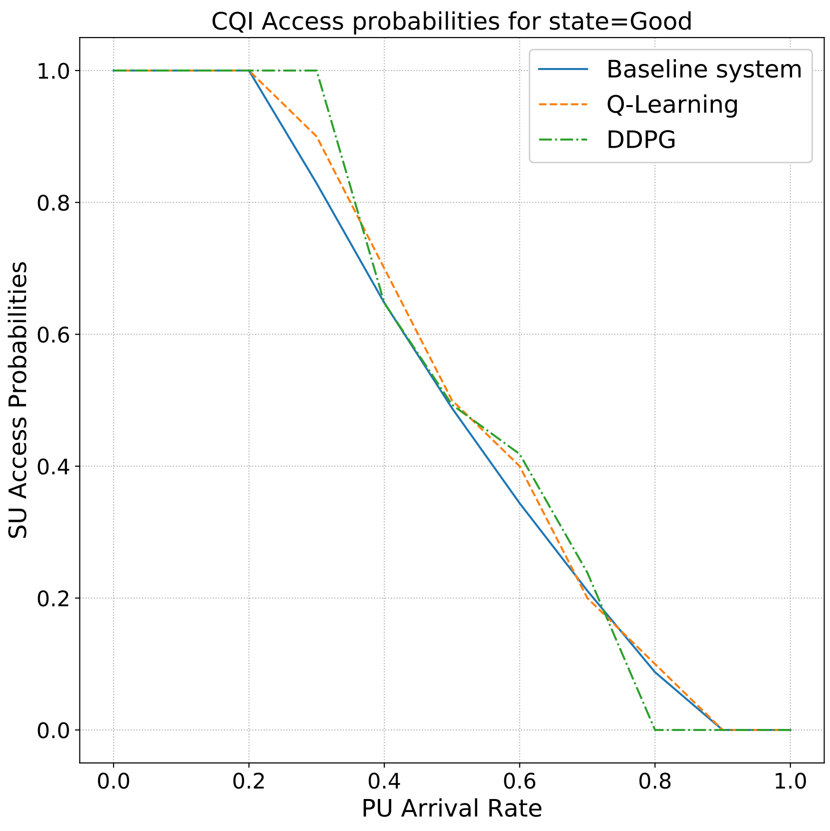

FIGURE 3.18: SU Access probabilities for CQI in the Good state ($p_B = 0.3$,$p_G = 0.9$,$\zeta_B = 0.125$,$\zeta_G = 0.875$,$p_D = 0.9$,$p_F = 0.1$)

zero PU arrival rate, the throughput is limited by the false alarm probability as the PU queue is always empty in this case, which is $1 - p_F = 0.9$. Also, at high arrival rates, the throughput converges to the steady-state probability of the channel being in the bad state, which is 0.125. However, adding HD sensing enhanced the throughput compared to the ARQ model, where no sensing was added, by decreasing collision between the PU and the SU. For instance, at 0.6 arrival rate Q-learning $r = 50$ algorithm in the ARQ system achieved 0.10445, while it achieved with $r = 10$, 0.18104 in the CQI.

Fig. 3.18 shows the SU access probabilities for the proposed algorithms in the CQI feedback scheme at the good state channel with the old method in [50], Q-learning, and DDPG algorithms. The closed-form expression for the access probabilities of the old method is shown in equation (3.18). The three algorithms shows close performance. However, DDPG shows slight advantage in the throughput as shown in Fig. 3.17. For instance, at 0.4 arrival rate, Q-learning and DDPG accessed with probabilities 0.7, 0.6475508 with throughput 0.30070763 for Q-learning, and 0.30808 for DDPG. Moreover, at 0.6 arrival rate, Q-learning and DDPG accessed with probabilities 0.4, 0.41840127, with throughput 0.18117183, and 0.19138. This minor gain can be crucial in applications that give high priority for every SU's timeslot value and success access.

Chapter 4

Hybrid (ARQ-CQI) feedback-based access scheme

In this chapter, A CR system is introduced whereby the SU utilizes both ARQ and CQI (Hybrid) PU feedbacks in the access scheme. Based on the SU spectrum HD sensing outcome, and PU feedback, it randomly accesses the PU channel. As a solution for the Hybrid POMDP problem, we propose the two RL model-free algorithms to teach the SU the access probabilities, which are Q-Learning, and DDPG.

4.1 Hybrid feedback Hard Detection (HD) based access scheme

The CR system's hybrid access scheme is one in which the SU combines both CQI and ARQ feedbacks from the PU to identify the probability of accessing the channel. When a bad CQI is received with any ARQ feedback, this means the channel state in the next time slot will be bad to deliver any message upon. Consequently, the PU will refrain from accessing the channel, and the SU will access with probability one. On the other hand, when a good CQI is received with (No-FB) or (ACK) ARQ feedback, the SU will transmit its packet in the next time slot, depending on the arrival rate, after applying HD sensing. However, When there is a (good) CQI feedback observed with (NACK) ARQ feedback, the SU will not access the channel as the PU will retransmit its packet in the next time slot. The hybrid mechanism has proved to show better performance over the other approaches where either none of ARQ or CQI is utilized or only of them is used at one time [24].

4.1.1 Channel model

We consider a Hybrid system that has a channel model, as shown in subsection 3.2.1.

4.1.2 System model

We consider a CR system has the system model, as shown in subsection 3.1.1.

4.1.3 PU queue model

We present three-dimensional Markov chain model for our PU queue as shown in Fig. 4.1. The state-space of this Markov chain, which is the set of values that the chain is allowed to take, is given by
$S = \{(K, D, T) : K = 0, 1, 2, ..., D \in \{F, R\}, T \in \{G, B\}\}$. K denotes the number of packets present in the PU's queue, D is the ARQ feedback, which contains first transmission F and re-transmission R request feedbacks. F means the first transmission of the PU packet that is at the head of the queue. When the first transmission of the packet fails, it is retransmitted, denoted by R. So R is the re-transmission of the packet present at the head of the queue. T represents the CQI feedback, where G and B denote the PU good channel and bad channel state [24].

The transitions probabilities between different states are derived as follows: Transition from (K, F, G) to $(K - 1, F, G), K > 0$: this transition occurs according to the following equation: $Pr(X(n+1) = (K-1, F, G)X(n) = (K, F, G))$= Pr((no new packet arrives at the PU queue) (SU does not detect the PU presence and decides not to access the channel) (the channel in the next time slot remains in the good state)) Pr(no new packet arrives at the

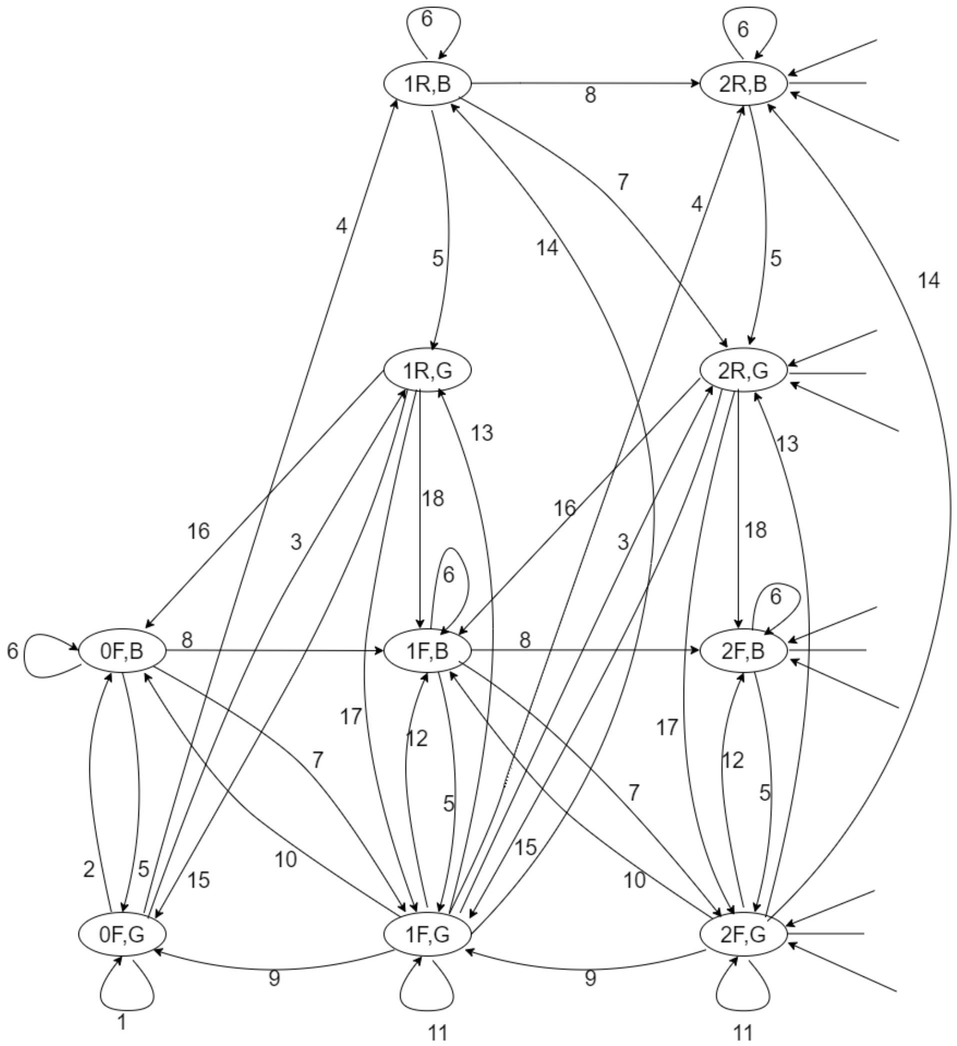

FIGURE 4.1: Three-dimensional MC model for the PU queue

PU queue) (SU detects the PU presence))= $(1-\lambda_p)((1-a_s)(1-p_d)+p_d)p_g$ [24].

Transition from (K,F,G) to $(K-1,F,B), K>0$:it is the same as the previous transition but p_g replaced by $(1-p_g)$. Therefore the transition probability equals to $(1-\lambda_p)((1-a_s)(1-p_d)+p_d)(1-p_g)$. Transition from (K,R,G) to (K,F,G), $K>0$: as the MC is in (K,R,G) the packet at the head of the queue will be transmitted successfully with probability 1. The transition in this case occurs according to this equation: $Pr(X(n+1)=(K,F,G)|X(n)=(K,R,G))$= Pr((new packet arrives at the PU queue) (the channel in the next time slot remains in the good state))= $\lambda_p p_g$. All remaining transition probabilities can be extracted from Fig. 4.1 and Table 4.1 [24].

TABLE 4.1: Transition probabilities of the Markov Chain for the ARQ-CQI feedback HD system.

Number	Transition Probability
1	$[(1-\lambda_p)+\lambda_p((1-a_s)(1-p_d)+p_d)]p_g$
2	$[(1-\lambda_p)+\lambda_p((1-a_s)(1-p_d)+p_d)](1-p_g)$
3	$\lambda_p a_s(1-p_d)p_g$
4	$\lambda_p a_s(1-p_d)(1-p_g)$
5	$(1-\lambda_p)(1-p_B)$
6	$(1-\lambda_p)p_B$
7	$\lambda_p(1-p_B)$
8	$\lambda_p p_B$
9	$(1-\lambda_p)[(1-a_s)(1-p_d)+p_d]p_g$
10	$(1-\lambda_p)[(1-a_s)(1-p_d)+p_d](1-p_g)$
11	$\lambda_p[(1-a_s)(1-p_d)+p_d]p_g$
12	$\lambda_p[(1-a_s)(1-p_d)+p_d](1-p_g)$
13	$(1-\lambda_p)a_s(1-p_d)p_g$
14	$(1-\lambda_p)a_s(1-p_d)(1-p_g)$
15	$(1-\lambda_p)p_g$
16	$(1-\lambda_p)(1-p_g)$
17	$\lambda_p p_g$
18	$\lambda_p(1-p_g)$

4.1.4 Transition states

State transitions in the Hybrid system consists of six states, No-FB (bad), No-FB (good), ACK (bad), ACK (good), NACK (bad), and NACK (good). As shown in Table 4.2 and 4.3, there are two tables, one for arrival and no arrival cases. In no arrival cases, the queue is always zero. Also, the possible states are only (No-FB good or bad), and (ACK good) feedbacks. If the SU receives (No-FB, good), (No-FB, bad), (ACK, good), or (ACK, bad), it gets a negative reward in case it did not access as it loses opportunity. However, it will receive a positive reward, if it accessed and the next state will be (No-FB good) or (No-FB bad) [24].

TABLE 4.2: Hybrid feedback for no arrival cases

q(t-1)	q(t) after arrival	S(t)	a(t+1)	S(t+1)	r(t+1)	q(t+1) because of action
No Arrival Cases						
0	0	1 = No-FB (good)	0 = no access	0 = No-FB (bad),1 = No-FB (good)	-1	0
0	0	1 = No-FB (good)	1 = access	0 = No-FB (bad),1 = No-FB (good)	1	0
0	0	1 = No-FB (bad)	0 = no access	0 = No-FB (bad),1 = No-FB (good)	-1	0
0	0	1 = No-FB (bad)	1 = access	0 = No-FB (bad),1 = No-FB (good)	1	0
1	0	3 = ACK (good)	0 = no access	0 = No-FB (bad),1 = No-FB (good)	-1	0
1	0	3 = ACK (good)	1 = access	0 = No-FB (bad),1 = No-FB (good)	1	0
1	0	3 = ACK (good)	0 = no access	0 = No-FB (bad),1 = No-FB (good)	-1	0
1	0	3 = ACK (good)	1 = access	0 = No-FB (bad), 1 = No-FB (good)	1	0

For arrival cases, the queue can be higher than zero, with an infinite number of states. If the SU receives (No-FB bad), and it did not access, it gets a negative reward. Nevertheless, if it accesses, it will receive a positive reward, and the next state could be No-FB (good or bad). On the other hand, if (No-FB good) is received, the SU gets a positive reward if it did not access, the next state is ACK (good or bad), and the queue will decrease by one. However, if it accessed, it gets a negative reward, the next state is NACK (good or bad), and the queue remains the same.

When the SU receives (ACK bad), if it did not access, it gets a negative reward, and the next state is ACK (good or bad). However, if it accesses, it receives a positive reward, and the next state is ACK (good or bad). On the other hand, if (ACK good) is received, the SU gets a positive reward if it did not access, the next state is ACK (good or bad), and the queue will decrease by one. However, if the SU accessed, it gets a negative reward, the next state is NACK (good or bad), and the queue remains the same.

When the SU receives (NACK bad), if it did not access, it gets a negative reward, and the next state is Nack (bad or good). Nevertheless, if it accesses, it gets a positive reward, and the next state is NACK (good or bad). On the other hand, if (NACK good) is received, the SU should back off, and it gets a positive reward if it did not access, the next state is ACK (good or bad), and the queue decrease by one. However, if the SU accessed, it gets a negative reward, the next state is NACK (good or bad), and the queue remains the same.

The next state is (bad or good), depending on the transition probability condition to stay in the current channel state or to change. For example, if the current channel state is bad, then the next state is bad if a random number is less than the probability of being in the bad channel, which is 0.3 and 0.9 respectively in the good channel, else it switches to the other channel condition.

TABLE 4.3: Hybrid feedback for arrival cases

q(t-1)	**q(t)** **(after arrival)**	**S(t)**	**a(t+1)**	**S(t+1)**	**r(t+1)**	**q(t+1)** **(because of action)**
Arrival Cases						
any value of q	q	0 = No-FB (bad)	0 = no access	0 = No-FB (bad), 1 = No-FB (good)	-1	q
any value of q	q	0 = No-FB (bad)	1 = access	0 = No-FB (bad) 1 = No-FB (good)	1	q
any value of q	q+1	0 = No-FB (bad)	0 = no access	0 = No-FB (bad) 1 = No-FB (good)	-1	q+1
any value of q	q+1	0 = No-FB (bad)	1 = access	0 = No-FB (bad) 1 = No-FB (good)	1	q+1
Greater than or equal to 1	q	1 = No-FB (good)	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q-1
Greater than or equal to 1	q	1 = No-FB (good)	1 = access	4-Nack (bad), 5 = Nack (good)	-1	q
any value of q	q+1	1 = No-FB (good)	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q
any value of q	q+1	1 = No-FB (good)	1 = access	4-Nack (bad), 5 = Nack (good)	-1	q+1
any value of q	q	2=ACK (bad)	0 = no access	0 = No-FB (bad), 1 = No-FB (good)	-1	q
any value of q	q	2=ACK (bad)	1 = access	0 = No-FB (bad) 1 = No-FB (good)	1	q
any value of q	q+1	2=ACK (bad)	0 = no access	0 = No-FB (bad) 1 = No-FB (good)	-1	q+1
any value of q	q+1	2=ACK (bad)	1 = access	0 = No-FB (bad) 1 = No-FB (good)	1	q+1
Greater than or equal to 1	q	3 = ACK (good	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q-1
Greater than or equal to 1	q	3 = ACK (good	1 = access	4-Nack (bad), 5 = Nack (good)	-1	q
any value of q	q+1	3 = ACK (good)	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q
any value of q	q+1	3 = ACK (good)	1 = access	4-Nack (bad), 5 = Nack (good)	-1	q+1
Greater than or equal to 1	q	4 = NACK (bad)	0 = no access	0 = No-FB (bad), 1 = No-FB (good)	-1	q
Greater than or equal to 1	q	4 = NACK (bad)	1 = access	0 = No-FB (bad), 1 = No-FB (good)	1	q
Greater than or equal to 1	q+1	4 = NACK (bad)	0 = no access	0 = No-FB (bad), 1 = No-FB (good)	-1	q+1
Greater than or equal to 1	q+1	4 = NACK (bad)	1 = access	0 = No-FB (bad), 1 = No-FB (good)	1	q+1
Greater than or equal to 1	q	5 = NACK (good)	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q-1
Greater than or equal to 1	q	5 = NACK (good)	1 = access	4-NACK (bad), 5 = NACK (good)	-1	q
Greater than or equal to 1	q+1	5 = NACK (good)	0 = no access	2=ACK (bad), 3 = ACK (good)	1	q
Greater than or equal to 1	q+1	5 = NACK (good)	1 = access	4-NACK (bad), 5 = NACK (good)	-1	q+1

4.1.5 POMDP environment framework

We consider a CR system with a POMDP environment framework, as shown in subsection 3.1.3 and 3.2.4.

4.1.6 POMDP MAC Policy

We consider a CR system that has the POMDP MAC Policy, as shown in subsection 3.1.4.

4.1.7 RL algorithms

We propose two RL algorithms to solve the Hybrid POMDP problem, Q-Learning as shown in subsection 3.1.6 and DDPG in subsection 3.1.8.

4.1.8 Simulation models

We compare our result with the conventional method which are the baseline approach.

- Baseline system The HD-Hybrid system drives a closed form expression based on a steady state distribution calculation of three dimensional MC model, given the know the arrival rate. The optimum access probability is shown in equation (4.1.8). The closed form expression for the throughput is shown in equation (3.29). Also, the perfect sensing curve is shown in equation (3.30) where the SU access whenever the PU's queue is empty and in the bad channel state [23].

$$a_s^* = (\lambda_p P_B - P_B - 2\lambda_p + \lambda_p P_G + 1)/(4\lambda_p - 4\lambda_p P_D - 2\lambda_p P_B - 2\lambda_p P_G + 2\lambda_p P_D P_B + 2\lambda_p P_D P_G),$$

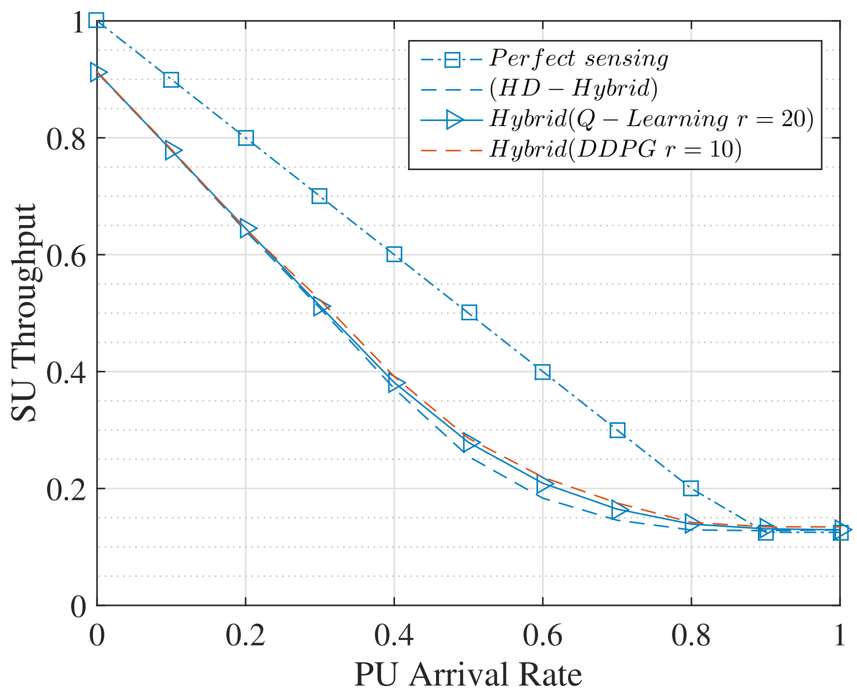

FIGURE 4.2: Hybrid feedback with RL (Q-Learning, DDPG) vs HD-Hybrid [23]

4.1.9 Hybrid Performance results

Finally, Fig. 4.2 compares the SU throughput for three access schemes, namely the baseline system approach in [23], the proposed Q-Learning with $r = 20$, and DDPG with $r = 10$ algorithm. At zero PU arrival rate, the throughput is 0.9, limited by the probability of detection and false alarm. Similarly, at arrival rate of 1, the throughput is 0.125, which is the probability of the PU being in the bad state. We use the same simulation parameters of sensing as Fig. 3.17. The DDPG-based approach yields the best performance with minor gain over Q-learning algorithm. For example, at 0.6 PU arrival rate, the SU access probability learned by Q-Learning in the (ACK and good) feedback is 0.5, which achieved throughput 0.208989. On the other hand, with DDPG, the learned access probability is 0.44 which achieved better throughput 0.21926. Therefore, allowing continuous action space in the case of DDPG can, in general, result in higher rewards. Also, The hybrid feedback system achieved higher throughput gain than using ARQ or CQI feedback alone. For example, at 0.6 arrival rate Q-learning

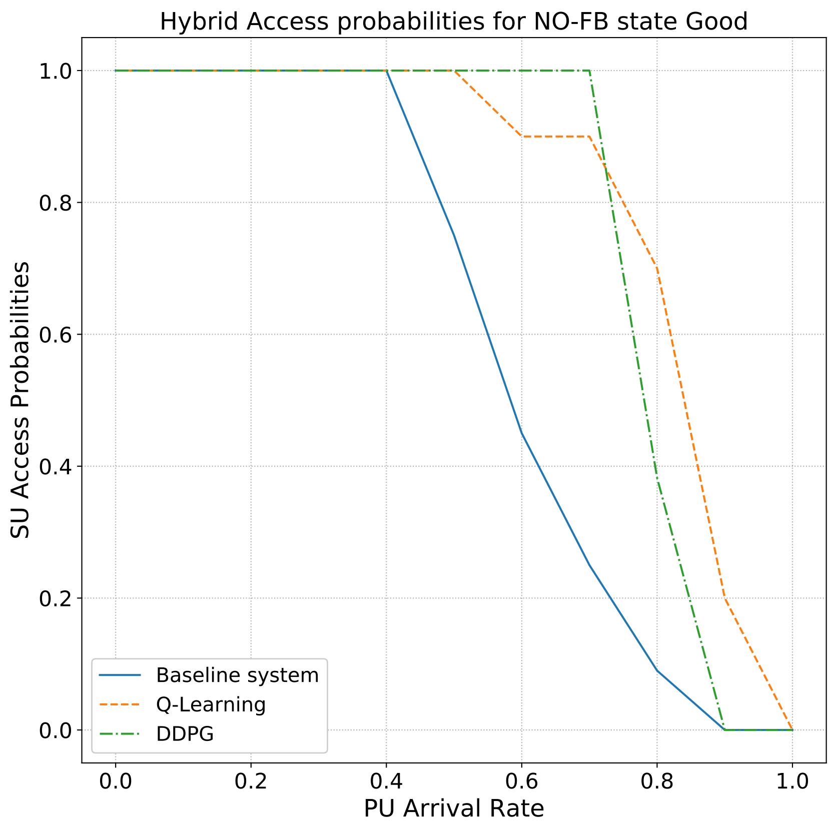

FIGURE 4.3: SU Access probabilities for Hybrid in the No-FB-Good state ($p_B = 0.3$,$p_G = 0.9$,$\zeta_B = 0.125$,$\zeta_G = 0.875$,$p_D = 0.9$,$p_F = 0.1$)

$r = 10$ algorithm in the CQI system achieved 0.18104, while it achieved with $r = 10$, 0.208989 in the hybrid.

Fig. 4.3 compares the access probabilities of the three proposed algorithms in the (No-FB and good state); namely, the baseline method in [24] using closed-form expression in equation (4.1.8), Q-Learning, and DDPG. In the throughput graph, both algorithms Q-learning and DDPG showed almost the same result with a slight advantage for DDPG. However, the throughput gap was wider between both algorithms and the old method, the same as in the access probability graph, which explains the advantage for using greedy RL algorithms besides accessing in the No-FB and the ACK state. DDPG algorithm accessed with probability one untill 0.7 arrival rate, then started to decrease as the arrival rate of the PU increases. However,

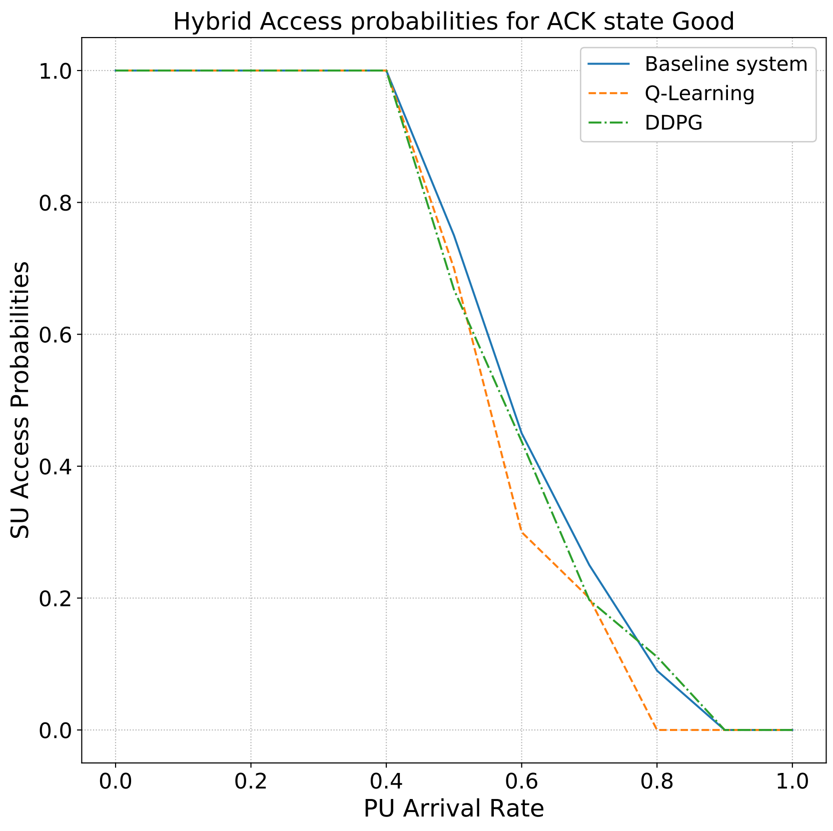

FIGURE 4.4: SU Access probabilities for Hybrid in the (ACK-Good state) $(p_B = 0.3, p_G = 0.9, \zeta_B = 0.125, \zeta_G = 0.875, p_D = 0.5, p_F = 0.1)$

that did not affect the QoS of the PU. For example at 0.8 arrival rate, Q-learning accesses with probability 0.7, and 0 in the (ACK and good state) achieving throughput 0.120902. However, DDPG accesses with probability 0.3822977 with throughput 0.14188.

Fig. 4.4 shows the Hybrid system's access probabilities in the (ACK and Good state). It compares Q-learning, DDPG, and the old method. (ACK and good state) showed less tendency toward accessing as compared to (No-FB and good state) due to the SU's sensing behavior to predict the PU availability in the next time slot. Similarly, the (ACK state) compared to the (No-FB state) in the ARQ systems. DDPG throughput showed slightly better performance due to its approximation access probability number out of the continuous action space, which can result in more successful access decisions.

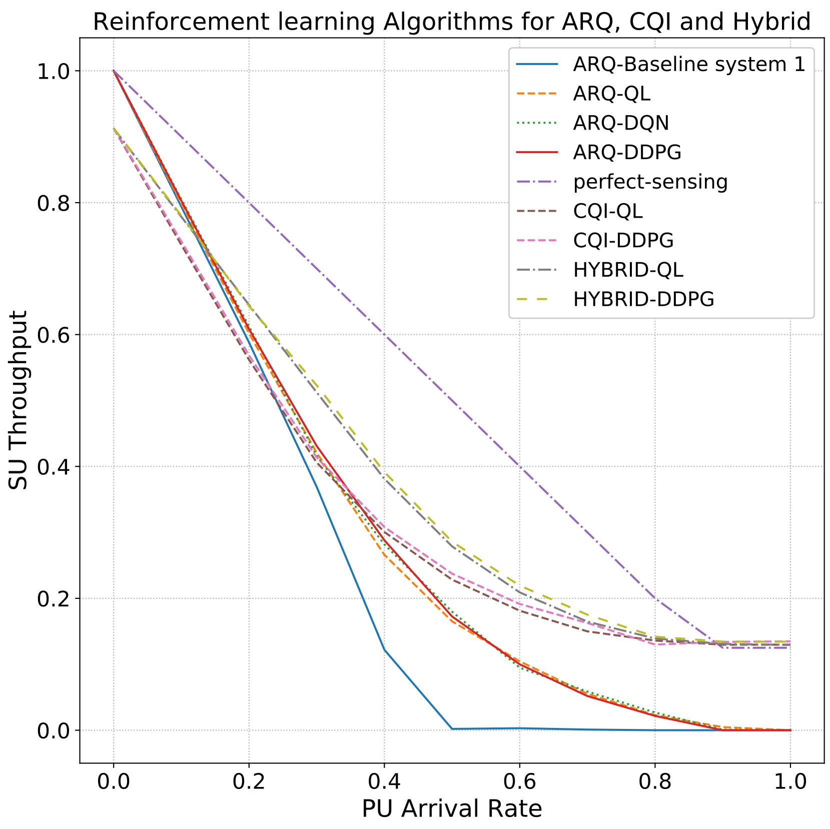

FIGURE 4.5: All RL algorithms for all PU feedback systems

Fig. 4.5 shows the comparison of the SU throughput achieved by our best proposed RL algorithms, namely Q-learning, DQN, and DDPG, used in all feedbacks systems ARQ, CQI, and Hybrid. CQI showed advancement over the ARQ since, in the bad channel, the SU knows that it will always send its packet. Moreover, after adding a HD sensing technique, the performance enhanced by making less collision with the primary user through the probability of detection and false alarm. Furthermore, the Hybrid system surpasses both the ARQ and the CQI by combining both feedbacks beside using HD sensing technique. For example, at 0.6 arrival rate, DDPG achieved 0.10732 throughput in the ARQ model, 0.19138 in the CQI, and 0.208989 in the Hybrid model. RL algorithms surpassed all the conventional methods by exploring and exploiting an unknown environment.

In the end, RL (Q-learning, DQN, and DDPG) algorithms have shown better throughput than conventional methods that require known parameters of the environment, such as the arrival rate and transition probabilities. Exploring and exploiting the environment devises an optimal strategy to learn the best action in each state, which is crucial in the absence of prior knowledge of the system. Being an AI system, RL proved to learn independently in any stochastic environment.

Chapter 5

Conclusions and future work

5.1 Conclusions

In this thesis, we consider the problem of designing the SU's access scheme in a cognitive radio system by exploiting the available PU's feedback information in the form of ARQ or/and CQI feedback to improve sensing. We consider three systems; one in which the SU has access only to the ARQ feedback, one in which the SU has access only to the CQI feedback, and one which the SU has access to both the ARQ and CQI feedback. The problem of the SU's access decision is modeled as a POMDP, which is solved using different RL-based approaches, namely, Q-Learning, DQN, and DDPG. Contrary to prior work, all of the RL based approaches assumed minimal knowledge about the PU activities to learn about the enviroment. Q-learning and DQN showed similar performance as they relied on discretizing the action space, which limits the access decisions. However, DDPG succeeded in fully exploring the whole continuous action space to make better access decisions, which resulted in minor throughput gain. This minor gain is crucial for the SU to make successful access attempts on the channel. For example, gaining access during congestion to one-time slot could deliver a message or part of a speech, which is vital in a battlefield, business world, or in emergency cases.

5.2 Future directions

We list a few potential recommendations for future studies to expand our current study.

- A multi-users model could be applied for PUs and SUs in CR in which one SU can access the unused channel of multiple PUs, hence, gaining more throughput.
- Soft sensing detection can be applied instead of Hard sensing to improve performance.
- Another direction is to extend the RL algorithm to consider other PU feedback information, or the history to better estimate PU's activity. We only studied ARQ, CQI, and Hybrid feedback in the RL based access scheme.
- Another direction is to try exploring different sensing approaches. It should be noted that the spectrum sensing approaches, as energy sensing, in this thesis can be used on top of any other PU sensing scheme and are always guaranteed to result in a performance gain.

Bibliography

[1] Sung Jang et al. "Reinforcement learning-based dynamic band and channel selection in cognitive radio ad-hoc networks". In: *EURASIP Journal on Wireless Communications and Networking* 2019 (Dec. 2019). DOI: `10.1186/s13638-019-1433-1`.

[2] Ridhima and Avtar Buttar. "Fundamental Operations of Cognitive Radio: A Survey". In: Feb. 2019, pp. 1–5. DOI: `10.1109/ICECCT.2019.8869190`.

[3] M. Olaleye, K. Dahal, and Z. Pervez. "A Hybrid Intelligence-Based Cognitive Engine". In: *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. 2019, pp. 258–262. DOI: `10.1109/CONFLUENCE.2019.8776899`.

[4] *Spectrum Collaboration Challenge*. `https://archive.darpa.mil/sc2/`. [Online; accessed 17-July-2020]. 2020.

[5] Arun Agarwal, Gourav Misra, and Kabita Agarwal. "The 5th Generation Mobile Wireless Networks- Key Concepts, Network Architecture and Challenges." In: *American Journal of Electrical and Electronic Engineering* 3 (Mar. 2015), pp. 22–28. DOI: `10.12691/ajeee-3-2-1`.

[6] M. Sherman et al. "IEEE Standards Supporting Cognitive Radio and Networks, Dynamic Spectrum Access, and Coexistence". In: *IEEE Communications Magazine* 46.7 (2008), pp. 72–79.

[7] Anita Garhwal and Partha Pratim Bhattacharya. "A survey on dynamic spectrum access techniques for cognitive radio". In: *arXiv preprint arXiv:1201.1964* (2012).

[8] S. A. Attalla et al. “Soft-Sensing CQI Feedback-Based Access Scheme in Cognitive Radio Networks”. In: *IEEE Transactions on Cognitive Communications and Networking* 4.3 (2018), pp. 486–499.

[9] A. M. Arafa et al. “A feedback-soft sensing-based access scheme for cognitive radio networks”. In: *IEEE Transactions on Wireless Communications* 12.7 (2013), pp. 3226–3237.

[10] Sassan Ahmadi. “The IEEE 802.16m Medium Access Control Common Part Sub-layer (Part I)”. In: Dec. 2011, pp. 169–279. ISBN: 9780123749642. DOI: 10.1016/B978-0-12-374964-2.10006-2.

[11] Ibrahim Maina. “Channel Quality Indicator Feedback in Long Term Evolution (LTE) System”. In: *IOSR Journal of Electronics and Communication Engineering* 9 (Jan. 2014), pp. 14–19. DOI: 10.9790/2834-09241419.

[12] Tam Nguyen, Frederic Villain, and Yann Guillou. “Cognitive Radio RF: Overview and Challenges”. In: *VLSI Design* 2012 (Feb. 2012). DOI: 10.1155/2012/716476.

[13] R. H. Puspita et al. “Reinforcement Learning Based 5G Enabled Cognitive Radio Networks”. In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 2019, pp. 555–558.

[14] L. Gavrilovska et al. “Learning and Reasoning in Cognitive Radio Networks”. In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 1761–1777.

[15] I.F. Akyildiz et al. “NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey”. In: *Computer Networks* 50.13 (2006), pp. 2127–2159.

[16] K. Eswaran, M. Gastpar, and K. Ramchandran. “Bits through ARQs: Spectrum Sharing with a Primary Packet System”. In: *2007 IEEE International Symposium on Information Theory*. 2007, pp. 2171–2175. DOI: 10.1109/ISIT.2007.4557542.

[17] S. Huang, X. Liu, and Z. Ding. “Distributed power control for cognitive user access based on primary link control feedback”. In: *IEEE International Conference on Computer Communications (INFOCOM).* San Diego, CA, 2010.

[18] M. Levorato, U. Mitra, and M. Zorzi. “Cognitive interference management in retransmission-based wireless networks”. In: *IEEE Transactions on Information Theory* 58.5 (2012), pp. 3023–3046.

[19] K.G. Seddik et al. “A feedback-based access scheme for cognitive radio systems”. In: *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC).* San Francisco, CA, 2011.

[20] A.M. Arafa et al. “A Feedback- Soft Sensing-Based Access Scheme for Cognitive Radio Networks”. In: *Wireless Communications, IEEE Transactions on* 12.7 (2013), pp. 3226–3237. ISSN: 1536-1276. DOI: 10.1109/TWC.2013.061413.120851.

[21] K. G. Seddik and A. A. El-Sherif. “A POMDP framework for cognitive MAC based on primary feedback exploitation”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP).* 2014, pp. 1281–1285. DOI: 10.1109/GlobalSIP.2014.7032329.

[22] S. A. Attalla et al. “Soft-Sensing CQI Feedback-Based Access Scheme in Cognitive Radio Networks”. In: *IEEE Transactions on Cognitive Communications and Networking* 4.3 (2018), pp. 486–499. DOI: 10.1109/TCCN.2018.2826539.

[23] S. A. Attalla et al. “Hybrid ARQ-CQI Feedback-Based Access Scheme in Cognitive Radio Networks”. In: *IEEE Transactions on Cognitive Communications and Networking* (2019), pp. 1–1.

[24] S. A. Attlla et al. “Hybrid Feedback-Based Access Scheme for Cognitive Radio Systems”. In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–7. DOI: 10.1109/GLOCOM.2017.8254557.

[25] Alex M. Andrew. “Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii + 322 pp, ISBN 0-262-19398-1, (hardback, £31.95)”. In: *Robotica* 17 (1999), pp. 229–235.

[26] Yuxi Li. “Deep Reinforcement Learning: An Overview”. In: *ArXiv* abs/1701.07274 (2017).

[27] Junta Wu and Huiyun Li. “Deep Ensemble Reinforcement Learning with Multiple Deep Deterministic Policy Gradient Algorithm”. In: *Mathematical Problems in Engineering* 2020 (2020), pp. 1–12.

[28] Vincent François-Lavet et al. “An Introduction to Deep Reinforcement Learning”. In: *CoRR* abs/1811.12560 (2018). arXiv: `1811.12560`. URL: `http://arxiv.org/abs/1811.12560`.

[29] Shaked Zychlinski. *The Complete Reinforcement Learning Dictionary*. `https://towardsdatascience.com/the-complete-reinforcement-learning-dictionary-e16230b7d24e`. Feb. 2019.

[30] Nadine Abbas and Karim Ahmad. “Recent advances on artificial intelligence and learning techniques in cognitive radio networks”. In: *Eurasip Journal on Wireless Communications and Networking* 2015 (Dec. 2015). DOI: `10.1186/s13638-015-0381-7`.

[31] Euge Inzaugarat. *Understanding Neural Networks: What, How and Why?* `https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31#:~:text=Briefly%2C%20a%20neural%20network%20is,state%20responses%20to%20external%20inputs.`. Oct. 2018.

[32] Faris B. Mismar and Brian L. Evans. “Deep Q-Learning for Self-Organizing Networks Fault Management and Radio Performance Improvement”. In: *CoRR* abs/1707.02329 (2017). arXiv: `1707.02329`. URL: `http://arxiv.org/abs/1707.02329`.

[33] Yonghua Wang et al. “A survey of dynamic spectrum allocation based on reinforcement learning algorithms in cognitive radio networks”. In: *Artificial Intelligence Review* 51 (June 2018). DOI: 10.1007/s10462-018-9639-x.

[34] Markus Buchholz. *Deep Reinforcement Learning. Deep Deterministic Policy Gradient (DDPG) algorithm.* https://medium.com/@markus.x.buchholz/deep-reinforcement-learning-deep-deterministic-policy-gradient-ddpg-algoritm-5a823da91b43. Mar. 2019.

[35] Ishan Jindal et al. “Optimizing Taxi Carpool Policies via Reinforcement Learning and Spatio-Temporal Mining”. In: Dec. 2018, pp. 1417–1426. DOI: 10.1109/BigData.2018.8622481.

[36] Jits Schilperoort et al. “Learning to Play Pac-Xon with Q-Learning and Two Double Q-Learning Variants”. In: Sept. 2018. DOI: 10.1109/SSCI.2018.8628782.

[37] Neil Hosey et al. “Q-learning for cognitive radios”. In: *Proceedings of the China-Ireland Information and Communications Technology Conference (CIICT 2009). ISBN 9780901519672*. National University of Ireland Maynooth. 2009.

[38] Avirup Das et al. “Q-Learning Based Co-Operative Spectrum Mobility in Cognitive Radio Networks”. In: Oct. 2017, pp. 502–505. DOI: 10.1109/LCN.2017.80.

[39] Hao Jiang et al. “An Improved Sarsa() Reinforcement Learning Algorithm for Wireless Communication Systems”. In: *IEEE Access* PP (Aug. 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2935255.

[40] J. Liu et al. “DeepNap: Data-Driven Base Station Sleeping Operations Through Deep Reinforcement Learning”. In: *IEEE Internet of Things Journal* 5.6 (2018), pp. 4273–4282. ISSN: 2372-2541. DOI: 10.1109/JIOT.2018.2846694.

[41] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[42] Andy. *Reinforcement learning tutorial using Python and Keras*. https://adventuresinmachinelearning.com/reinforcement-learning-tutorial-python-keras/?fbclid=IwAR1-b4B4XUiGG2ALE1S--MovSb6LpUV0Cv39yO3FCxgP6_NWlV2-OC4HBOCR0. [Online; accessed 14-jan-2020]. 2020.

[43] Alex Potapov and M. Ali. "Convergence of reinforcement learning algorithms and acceleration of learning". In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 67 (Mar. 2003), p. 026706. DOI: 10.1103/PhysRevE.67.026706.

[44] S. Wang et al. "Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks". In: *IEEE Transactions on Cognitive Communications and Networking* 4.2 (2018), pp. 257–265.

[45] H. Chang et al. "Distributive Dynamic Spectrum Access Through Deep Reinforcement Learning: A Reservoir Computing-Based Approach". In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 1938–1948.

[46] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[47] Vitaly Bushaev. *Adam — latest trends in deep learning optimization*. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c. [Online; accessed 14-jan-2020]. 2018.

[48] Yi-Han Xu et al. "Deep Deterministic Policy Gradient (DDPG)-Based Resource Allocation Scheme for NOMA Vehicular Communications". In: *IEEE Access* PP (Jan. 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.2968595.

[49] S. Tseng et al. "Radio Resource Scheduling for 5G NR via Deep Deterministic Policy Gradient". In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2019, pp. 1–6.

[50] S Attalla et al. "Channel quality feedback-based access scheme for cognitive radio systems". In: *Proc. IEEE GLOBECOM*. 2016.