

**A DEEP REINFORCEMENT LEARNING
APPROACH FOR MINIMIZING AGE OF
INFORMATION IN EDGE CACHING FOR
COGNITIVE CITIES**

BY

ALA MOHAMMED ALHAMDAN

A Thesis Presented to the
COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

JANUARY 2024

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ALA MOHAMMED ALHAMDAN** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS**.

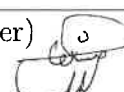
Thesis Committee

2 May 2024



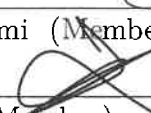
Dr. Yahya Ismail Osais (Adviser)

May 7, 2024



Dr. Tarek Rahil Sheltami (Member)

7 May 2024



Dr. Louai Al-Awami (Member)



Dr. Aiman Helmi El-Maleh
Department Chairman



Dr. Suliman Al-Homidan
Dean of Graduate Studies

Date



©Ala Mohammed Alhamdan
2024

Dedication

**This thesis is dedicated to my Father and my Mother, who were my rock
throughout**

Mr. and Mrs. Mohammed and Luluwah
and

My Brothers and Sisters

Tareq , Suliman , Abdullah , Muneerah , Khalid and Arwa.

ACKNOWLEDGMENTS

"Praise be to Allah, by whose grace good deeds are completed."

First and foremost, Alhamdulillah, I praise and thank Allah for giving me the opportunity, patience, and strength to complete my master's degree.

I would like to express my deep gratitude to my parents, Mrs. Luuwah and Mr. Mohammed, for their unlimited support, patience, and love.

I would like to thank my brothers and sisters for supporting and motivating me during my studies.

I extend my special thanks to my sisters Muneerah and Arwa. Their unwavering support, encouragement, and understanding were a constant source of motivation throughout this journey.

I owe a deep debt of gratitude to my advisor, Dr. Yahya Ismail Osais. His expertise, guidance, and insightful feedback have been instrumental in shaping this thesis. I appreciate the time and effort invested in mentoring me, fostering critical thinking, and providing invaluable direction to navigate the complexities of the research process.

I would like to thank my thesis committee Dr.Aiman Helmi El-Maleh , Dr.Tarek

Rahil Sheltam and Dr.Louai Al-Awami. The exchange of ideas and constructive discussions enriched my understanding and contributed significantly to the development of this work.

Moreover, my sincere appreciation goes to all the professors who have played a vital role in my academic journey. Each of them has imparted valuable knowledge and inspired a passion for continuous learning.

Finally, I extend my thanks to King Fahd University of Petroleum and Minerals for providing me with the opportunity to pursue higher education. The resources, academic environment, and support from the university have been crucial in shaping my academic and personal growth.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
ABSTRACT (ENGLISH)	xiv
ABSTRACT (ARABIC)	xvi
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	3
1.2 Problem statement	6
1.2.1 Research Questions	6
1.3 Contribution	6
1.4 Thesis overview	8
CHAPTER 2 BACKGROUND	10
2.1 Age of Information	10
2.2 Reinforcement Learning	12
2.3 Markov Decision Process	13
2.4 Q-learning Algorithm	14
2.5 Deep Reinforcement Learning	14
CHAPTER 3 LITERATURE REVIEW	16
3.1 Reinforcement Learning approach	16

3.2	Deep Reinforcement Learning approach	18
3.3	Cache Replacement Policies	19
3.4	Discussion	19
CHAPTER 4 SYSTEM MODELING		22
4.1	System model	22
4.2	MDP model	23
CHAPTER 5 SIMULATION		30
5.1	Environment	30
5.2	Q-Learning Algorithm	34
5.3	DqN Algorithm	35
5.4	Greedy Algorithm	39
5.5	Random Algorithm	39
CHAPTER 6 SYSTEM EVALUATION		41
6.1	System Evaluation Over Time	41
6.1.1	Environment Case 1	41
6.1.2	Environment Case 2	44
6.1.3	Environment Case 3	45
6.2	System Evaluation Under Changing Environment Parameters	48
6.2.1	Changing Probability of a successful transmission	48
6.2.2	Changing Probability of Energy Harvesting	50
6.3	Discussion	52
CHAPTER 7 CONCLUSION		54
7.1	Conclusion	54
7.2	Future Work	55
REFERENCES		56
VITAE		61

LIST OF TABLES

1.1	An example of the specific locations (states) of AGVs and their AoI in different time slots.	5
2.1	RL models.	13
4.1	Generalized MDP models and proposed MDP model.	25
4.2	Coverage Matrix of environment case 1	29
4.3	Coverage Matrix of environment case 2.	29
4.4	Coverage Matrix of environment case 3.	29
6.1	System Parameters.	41
6.2	Average on demand AoI of fully covered and Half covered environment case 1	44
6.3	Percentage of minimizing for fully covered and Half covered environment case 1	44
6.4	Average on demand AoI of fully covered and Half covered environment case 2	46
6.5	Percentage of minimizing for fully covered and Half covered environment case 2	46
6.6	Average on demand AoI of fully covered and Half covered environment case 3	46
6.7	Percentage of minimizing for fully covered and Half covered environment case 3	47

LIST OF FIGURES

1.1	The three IoT layers in the application domain of cognitive cities. . .	2
1.2	The general model of cognitive cities.	2
1.3	The AGVs move in an anti-clockwise direction, in addition, in zone one, the AGVs has to stay two time slots before moving. [1]	5
1.4	One to One relationship	8
1.5	Many to Many relationship	8
2.1	Components of AoI.	11
2.2	Sample path of AoI.	11
2.3	Markov Decision Process Steps.	14
2.4	The main difference between Q-Learning and Deep Q Network.	15
3.1	Comparison between the MDP models in the references.	21
4.1	System model.	23
4.2	Fully Covered Environment for case 1.	27
4.3	Fully Covered Environment for case 2.	27
4.4	Fully Covered Environment for case 3.	28
5.1	Environments case 1.	32
5.2	Environments case 2.	32
5.3	Environments case 3.	32
5.4	Flowchart for the simulation of the environment.	33
5.5	DqN diagram.	36
5.6	DRL architecture.	37

6.1	environment case 1 Fully Covered	43
6.2	environment case 1 half Covered	43
6.3	environment case 2 fully Covered	45
6.4	environment case 2 half Covered	45
6.5	environment case 3 fully Covered	47
6.6	environment case 3 half Covered	47
6.7	environment case 1 Changing Probability of a successful transmission DqN	49
6.8	environment case 1 Changing Probability of a successful transmission Q-Learning	49
6.9	environment case 1 Changing Probability of a successful transmission Greedy	50
6.10	environment case 1 Changing Probability of a successful transmission Random	50
6.11	environment case 1 Changing Probability that a sensor harvests one unit of energy DqN	51
6.12	environment case 1 Changing Probability that a sensor harvests one unit of energy Q-Learning	51
6.13	environment case 1 Changing Probability that a sensor harvests one unit of energy Greedy	52
6.14	environment case 1 Changing Probability that a sensor harvests one unit of energy Random	52

THESIS ABSTRACT

NAME: Ala Mohammed Alhamdan

TITLE OF STUDY: A Deep Reinforcement Learning Approach for Minimizing
Age of Information in Edge Caching for Cognitive Cities

MAJOR FIELD: Computer Networks

DATE OF DEGREE: January 2024

The economy of Saudi Arabia is being transformed into a digitally empowered economy. Smart cities, like NEOM, are at the core of this new economy. Services in smart cities will become cognitive. Such services will require a huge amount of timely data, which is produced and collected by a complex network of IoT devices. As a result, data management will become a real challenge in smart cities with cognitive services. Caching is an essential functionality in the architecture for smart city applications. Its purpose is to keep the data close to the application and thus offload the data management responsibility to the edge. However, cache capacity is limited and data in the cache can become stale and may not be relevant to the application anymore. Cognitive applications are sensitive to the age of data stored in the cache. Therefore, new sophisticated cache management techniques are needed to ensure the freshness

of data and its relevance to the application. In this paper, we aim to develop a new generalized model for cache management in cognitive cities. In addition, in order to ensure on-demand freshness of data, a Deep Reinforcement Learning (DRL) approach is used based on the new generalized model. Our simulation results show that the DRL algorithm can achieve a reduction of 11.2% in on-demand Age of Information (AoI) when compared with the standard Q-learning algorithm. Also, the DRL algorithm can achieve a reduction of 30% and 36% in on-demand AoI when compared to a greedy algorithm and random algorithm, respectively.

Keywords: Age of Information (AoI), data caching, Deep Reinforcement Learning (DRL), the Internet of Things (IoT), cognitive cities, and AoI-sensitive applications.

ملخص الرسالة

الاسم:

آلاء بنت محمد سليمان الحمدان

عنوان الدراسة:

التعليم المعزز والعميق لتقليل عمر البيانات للمدن الإدراكية

التخصص:

شبكات الحاسب الآلي

تاريخ الدرجة العلمية: يناير 2024

تزامنا مع انطلاقة رؤية 2030 في المملكة العربية السعودية والنقلة الفريدة من نوعها لتطوير المدن الذكية والإدراكية في نفس الوقت على سبيل المثال لا الحصر : مدينة نيوم في شمال غرب المملكة العربية السعودية تم تخصيص هذا البحث العلمي بعنوان التعليم المعزز والعميق في تقليل عمر البيانات للمدن الإدراكية لمواكبة التطور الحالي في المملكة العربية السعودية . المدن الذكية والإدراكية تنتج العديد من البيانات من خلال الحساسات المنتشرة فيها , لذا جودة البيانات ترتبط بوقت وصولها و من الممكن أن يؤثر وقت وصولها المتأخر إلى جودة اتخاذ القرارات. لذا في هذه الرسالة سنناقش أهمية جودة البيانات المستخرجة من الحساسات إلى وقت وصولها إلى مركز البيانات المصغر والمنتشر في المدن الذكية لمعالجتها باستخدام خوارزميات الذكاء الاصطناعي و التعلم بالتجربة. مع الوضع في عين الاعتبار مقياس الأداء في النظام وهو عمر المعلومات. يصنف عمر المعلومات بأنه مقياس جديد لجودة البيانات في اتخاذ القرارات لأي حالة في النظام المطبق في هذه الرسالة. تظهر النتائج المستخرجة من تطبيق خوارزميات الذكاء الاصطناعي تطور ملحوظ في التقليل من قيمة عمر المعلومات. في هذه الرسالة تعتمد الخوارزمية بشكل مباشر على الشبكات العصبية العميقة. وقد أظهرت الخوارزمية الجديدة المقترحة القدرة على تقليل قيمة عمر المعلومات مقارنة باستخدام الخوارزميه العشوائية و خوارزمية الشجرة. تظهر النتائج النهائية نقصان بنسبة احدى عشر فاصلة اثنان بالمئة عند استخدام خوارزمية التعليم العميق مقارنة باستخدام خوارزميه التعليم بالتجربة التي تعتمد على استخدام الجداول وبنسبة ثلاثون بالمئه مقارنة باستخدام الخوارزمية الشجرة ونسبة ستة وثلاثون مقارنة باستخدام الخوارزمية العشوائية.

CHAPTER 1

INTRODUCTION

Saudi Arabia is known as one of the leading countries in the oil industry. However, the country has an ambitious plan to diversify its economy. The leaders of the country launched multiple initiatives to achieve this goal. One of the most important initiatives is the next generation of smart cities, i.e., cognitive cities. The project described in [2] exemplifies this initiative.

In a cognitive city, the city understands what happens inside its environment and predicts what will happen next [3]. The IoT-based infrastructure of the city will provide the huge amount of data needed for these cognitive abilities. As a result, data management becomes an important issue in the operations of cognitive cities.

In addition to consuming a huge amount of data, applications in cognitive cities require their data to be up-to-date with a short window for expiration. Applications set a certain limit beyond which data expires. Cognitive operations in smart cities are useful only if the age of their input data is minimal. Therefore, the problem of managing the age of data in cognitive cities becomes very important.

Furthermore, data management will naturally lead to a hierarchy of caches. For

example, Figure 1.1 illustrates the three layers that are typically found in the architecture of applications in cognitive cities. The three layers are the cloud layer, the edge layer, and the IoT infrastructure layer. In this thesis, we are interested in managing the data caches storing sensor data on the edge. These data caches will be used to serve fresh data to the applications in layer 3. Sources in layer 1 (Figure 1.1) generate the data, and IoT devices deliver it to the caches in layer 2 (i.e., the edge).

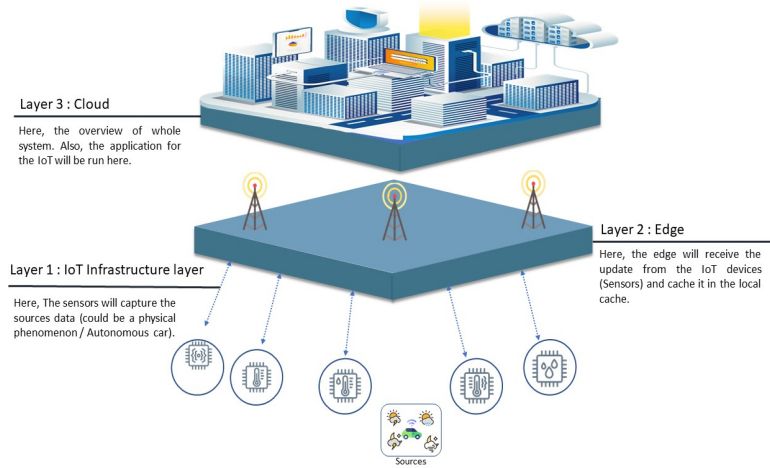


Figure 1.1: The three IoT layers in the application domain of cognitive cities.

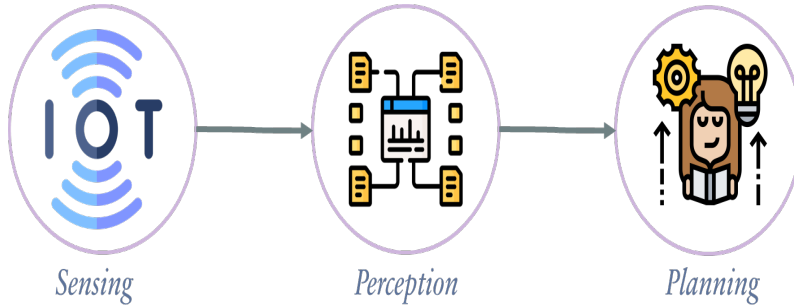


Figure 1.2: The general model of cognitive cities.

1.1 Motivation

The Internet of Things (IoT) refers to a network of interconnected devices that can communicate and share data with each other through the internet. In the context of the next example, automatic guided vehicles (AGVs) can be considered IoT devices. Sensors equip each AGV, and cameras monitor them, creating a system that continuously collects and shares data to optimize operations on the warehouse floor. The integration of IoT elements, such as cameras and AGVs, enables a dynamic and interconnected system that enhances the monitoring and control of warehouse operations.

The following motivating example is adapted from [1]. Consider a system of three Automatic Guided Vehicles (AGVs) monitored by a network of three cameras as shown in Figure 1.3. The floor is divided into four zones. Zones Z1, Z2, and Z4 are continuously monitored by cameras C1, C2, and C4, respectively. Zone Z3 is not monitored by a camera. The AGVs move on the floor in an anti-clockwise direction. Also, an AGV in zone Z1 has to stay for two time slots before moving to zone Z4. The purpose of the cameras is to detect the presence of the AGVs in the monitored zones. AGVs are distinguished by their colors as shown in Figure 1.3.

For the monitoring application, the state of the AGV represents its location at a specific point in time. For example, in time slot number one, zones one, two, and three, respectively, contain AGV numbers one, two, and three. The initial AoI for each AGV is one, one, and four, respectively. The decision in time slot one is to observe the AGVs using camera 1 (C1). This will result in the new AoI in time slot two, where only the AoI for AGV 1 is updated because the AGVs in zones Z2 and Z3 are not

covered by camera C1. As a result, AoI for AGV2 and AGV3 increase by one unit. In this case, the average AoI for the system in time slot 2 is 2.67. On the other hand, in slot number five, camera one observes both AGV number two and AGV number three, which are located in zone number one. The AoI of these two AGVs will be become one in the next time slot. However, camera one will not observe AGV number three, resulting in an increase of one in its AoI. Thus, the average system AoI is 1.33. Clearly, in this specific example, scheduling which camera should observe the AGVs in every time slot will have a significant impact on the average AoI of the system. The last row in Table 1.1 shows the variation in the average AoI for the system as cameras are selected for monitoring. Therefore, a minimum average AoI is crucial in applications that require a timely view of the operations on the floor.

To sum up, in this example, the AoI for every AGV represents the time elapsed since the last event of observing the AGV in the system. The average AoI for the system, however, is a measure of the quality of observability in the system. That is, as the average AoI for the system increases, observability in the system decreases and the monitoring application will not be able to tell even the approximate locations of the AGVs.

In this example, the trade-offs revolve around the scheduling decision of which camera should observe the Automatic Guided Vehicles (AGVs) in each time slot. The decision to use a specific camera affects the updating of AoI for each AGV. Opting for a particular camera may result in a decrease in the AoI for some AGVs and an increase for others. For instance, in time slot one, choosing Camera 1 (C1) updates the AoI for AGV 1 but increments it for AGV 2 and AGV 3. The challenge lies

Time slot	1	2	3	4	5	6
Location	(Z1,Z2,Z3)	(Z2,Z3,Z4)	(Z3,Z4,Z1)	(Z4,Z1,Z1)	(Z1,Z1,Z2)	(Z1,Z2,Z3)
Policy	C1	C4	C4	C1	C1	C1
AoI	(1,1,4)	(1,2,5)	(2,3,1)	(3,1,2)	(4,1,1)	(1,1,2)
Avg AoI	2	2.67	2	2	2	1.33

Table 1.1: An example of the specific locations (states) of AGVs and their AoI in different time slots.

in finding a scheduling policy that minimizes the average AoI for the entire system. Intelligent camera scheduling is essential to achieve this goal, considering the dynamic movement of AGVs across different zones and the impact of camera coverage on their AoI. Achieving a minimum average AoI is crucial, especially in applications requiring timely monitoring of operations on the warehouse floor.

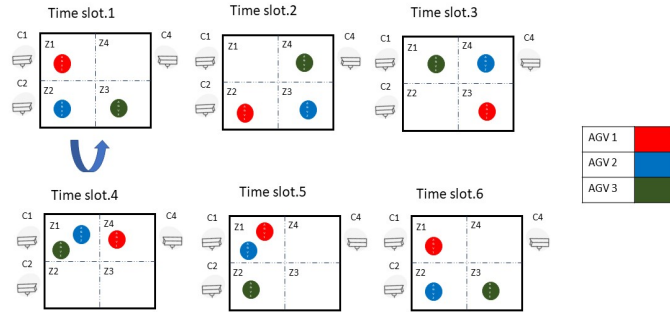


Figure 1.3: The AGVs move in an anti-clockwise direction, in addition, in zone one, the AGVs has to stay two time slots before moving. [1]

1.2 Problem statement

On-demand AoI measures the freshness of data upon request. Before the response is returned, the data has to be opportunistically refreshed using an intelligently selected sensor out of a group of potential sensors. These sensors cover sources in many-to-many relationships. The goal of the agent on the edge is to make sure that the average of the on-demand AoI of the returned vector of data is as small as possible. This is in contrast to the common approach in the literature, where the goal is to minimize the average AoI over the lifetime of the system.

1.2.1 Research Questions

The following are the research questions that we aim to answer:

1. What is the optimal policy that minimizes the average on-demand AoI using the Q-learning algorithm and the Deep Q Network (DQN) algorithm?
2. How effective are the newly obtained policies when they are compared with random and greedy policies?
3. What are the benefits of combining the two MDP models from [1] and [4] into one generalized model?

1.3 Contribution

On-demand AoI (od-AoI) was introduced in [5]. od-AoI is a measures the freshness of data upon request. That is, the requested data is refreshed before a response is

returned to the requesting application. This is in contrast to the common approach in the literature, where the goal is to minimize the average AoI over the lifetime of the system. As such, the data is always refreshed even if it is not requested.

In a system with a one-to-one relationship between sensors and sources, like the system in [5], on-demand data freshness is guaranteed provided that the sensor still energy in its battery for transmission. Therefore, the main trade-off will be between energy consumption and od-AoI. However, a third element of a potential triad is missing. This missing element is the coverage, which establishes many-to-many relationships between sensors and data sources figure 1.5. The issue of coverage in a system with many-to-many relationships between sensors and data sources was considered in [1]. In this new system configuration, the standard AoI was considered.

There is a need to study the od-AoI metric in an environment with many-to-many relationships between sensors and data sources. In such an environment, the data is opportunistically refreshed if the data source is covered by the sensor. Furthermore, the data needed to fulfill a request may come from different sources, which may not all be covered by the selected sensor. Clearly, in this case, the goal is to make sure that the average of the on-demand AoI of the returned vector of data is as small as possible.

In this paper, we make the following contribution:

1. Combine the models in [1] and [4] into one generalized model for studying the od-AoI in environments with many-to-many relationships between sensors and data sources and multi-target application requests.

2. Solve the new problem by using the deep reinforcement learning algorithm DqN to handle the large system state space.
3. Evaluate the obtained policy with other baseline policies using Monte Carlo simulations.

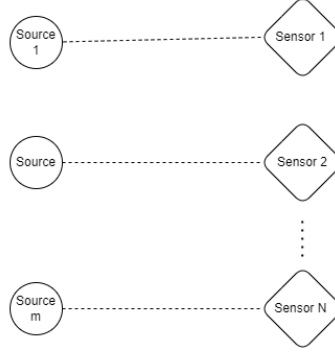


Figure 1.4: One to One relationship .

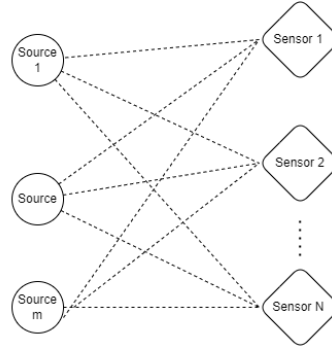


Figure 1.5: Many to Many relationship .

1.4 Thesis overview

The rest of the thesis is structured as follows:

The necessary background information is given in Chapter 2. The literature review is given in Chapter 3. The system model and system implementation are presented

in chapters 4 and 5, respectively. The system Evaluation is discussed in Chapter 6. Finally, the conclusion is given in Chapter 7 with recommendations for future work.

CHAPTER 2

BACKGROUND

In this chapter, we will give the necessary background that is needed to understand the thesis concepts. In the first section, we will discuss the age of information performance metric that is used to evaluate the system in this thesis. Section 2 covers Reinforcement Learning, Section 3 discusses the Markov Decision Process model, and Section 4 covers the Q-learning algorithm. Finally, deep reinforcement learning is in Section 5.

2.1 Age of Information

Age of Information (AoI) is a new performance metric in the Internet of Things (IoT) era [6]. The Age of Information was introduced in [7]. Unlike the delay, the AoI is about the quality of the data on the receiver side after it is received. Therefore, the age of information is used for measuring the freshness of data. Since we can find the AoI by assigning t_g the time when the data is gathered on the source side, t_a is the time when the information is received at the destination, and t is the endpoint when

the data is fully received, as it shows in 2.1 (see Equation 2.1),.

$$\Delta(t) = t_a - t_g \quad (2.1)$$

To consider the information as fresh, the state update between t and t_g must be lower than the maximum age allowed by the application. If the state update is higher, we consider the information to be stale. Thus, AoI is “time elapsed since the generation of the last received update message” [7]



Figure 2.1: Components of AoI.

Figure 2.2 illustrates one possible sample path of the AoI function. Over time, any new update will reset the age counter to zero (which is at times 1, 2, and 9). Between time 2 and 9, the age will keep increasing until it reaches the maximum age (which is 4 in this figure (example)). When it reaches the maximum age, the data will be unusable.

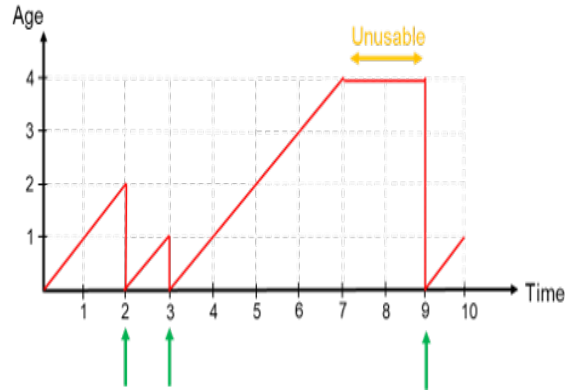


Figure 2.2: Sample path of AoI.

On-demand AoI is introduced in [5] . On-demand Age of Information (AoI) is a concept introduced to address the specific needs of IoT applications where information freshness becomes critical only upon the request of the end user. Unlike the traditional AoI, which is concerned with keeping information as fresh as possible at all times, on-demand AoI focuses on minimizing the age of information, specifically at the moments when an end user requests new data. This approach acknowledges that the relevance of information freshness may not be constant but rather dependent on the demand for that information by users. It represents a more nuanced and efficient way of managing information flow within IoT systems, especially in scenarios where conserving resources like bandwidth and energy is essential. The on-demand AoI is therefore critical for applications where users intermittently request updates, and the system aims to optimize the freshness of the information delivered in response to these requests, balancing the need for up-to-date information against the cost of providing it.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a part of artificial intelligence. RL learns what policy should be used and the best action to maximize the rewards. RL contains three elements: policy, reward, and model [8] . The three elements are defined as follows:

- Policy: Mapping the state of the environment to its corresponding action.
- Reward: The feedback from the environment after processing an action. (in the form of a number).

Num	RL	State	Action	Reward	Transition Probability	Online
1	MDP	✓	✓	✓	✓	×
2	Q-learning Algorithm	✓	✓	✓	×	✓
3	DRL	✓	✓	✓	×	✓

Table 2.1: RL models.

- Model: used for testing the behaviors of the environment for predicting the outcome of policy.

There are many models of RL. Table 2.1 shows the different models. Models 2–3 are used. online; in contrast, model 1 is offline, which makes it harder since it needs to collect the data before starting the model. On the other hand, models 2–3 are considered easier since they do not require a State Transition Probability. The main difference between models 2 and 3 is that in model 3, it required the action value function. More details about the models are discussed below.

2.3 Markov Decision Process

The Markov Decision Process (MDP) is a mathematical model for reinforcement. Learning problem. MDP focuses on decision-making using current knowledge about the system state, and it evolves based on the observations obtained from the environment. see figure 2.3. MDP contains four elements: state, action, state transition probability, and Rewards. The State Transitions Probability is the probability of an action with a state at a specific time slot will lead to the next state [8] .



Figure 2.3: Markov Decision Process Steps.

2.4 Q-learning Algorithm

The Q-learning algorithm is model-free reinforcement learning. The main goal of Q-learning is to find the optimal policy. To achieve this goal, the algorithm collects the action-value function in each iteration iteratively. The action-value function will estimate the good action for the environment from the given state and action. Consequently, in each time slot, the system in a certain state s_t will choose an action a_t .

The environment will run the selected action, and then it will store the reward that corresponds to performing the action. As a result, when the reward is high, the action will be considered a good selection. The state transition probabilities are not required, which makes it a special feature in the Q-learning algorithm [8] .

2.5 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) describes solving high-dimensional decision-making problems by combining deep learning and reinforcement learning. One of the benefits of combining deep learning and reinforcement learning is that it allows the reinforcement learning agent to discern the surrounding environment. Deep learn-

ing finds the low-dimensional features of the row data (like row sensor data, images, etc.). This property of deep learning will allow reinforcement learning to increase the state and action space of reinforcement learning. One of the major algorithms in DRL is the Deep Q Network (DQN). DQN is an enhanced Q-learning from reinforcement learning with deep neural networks. The main feature of DQN is that rather than mapping the input state by an action-state function (in Q-learning), the input will be mapped by a neural network [9] , [10] and [11] .Figure 2.4 shows the main difference between the Q-learning and Deep Q networks.

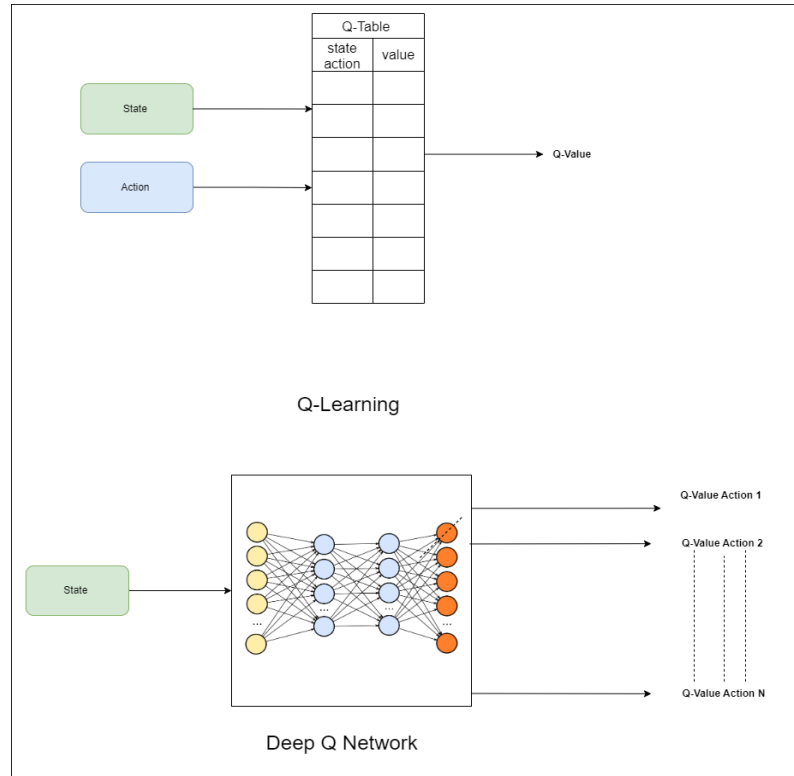


Figure 2.4: The main difference between Q-Learning and Deep Q Network.

CHAPTER 3

LITERATURE REVIEW

In this chapter, we will review papers that covered AoI as a performance metric, along with using the MDP framework. In the first section, the problem is solved by using the reinforcement learning approach; in section 2, by using the deep reinforcement learning approach; and in the third section, cache replacement policies.

3.1 Reinforcement Learning approach

In [5], the authors formulate the problem as an MDP model and solve it by using the Relative Value Iteration Algorithm. The goal is to reduce the average on-demand AoI for energy-harvesting sensors. Achieving the goal involves finding the optimal action on the edge node, which includes either commanding the sensor or not. The system contains three elements, which are the end user, the edge node, and energy harvesting sensors. In the MDP model, the AoI and the EH sensors are considered states. There are two actions in the MDP model, which are either to command the sensor to send a new update or to use the data stored in the cache. If the edge node

successfully receives the status update h_k from the sensor, the cost function assigns a cost of 1; otherwise, it assigns a cost equal to the minimum of AoI +1 and the maximum AoI (see Equation 3.1). The simulation result indicates that the proposed algorithm achieves a 30% reduction in average cost compared to the greedy policy and a 10% reduction compared to the pure Age of Information (AoI) method, which aims to minimize the average AoI at the edge node.

$$\Delta(t+1) = \begin{cases} 1, & \text{if } h_k = 1 \\ \min \{ \Delta_k(t) + 1, \Delta_{k,max} \}, & \text{if } h_k = 0 \end{cases} \quad (3.1)$$

Like [5] , the authors in [4] discussed the issues of finding the optimal action on the edge node by either commanding the sensor or not. The main difference is that in [4] , the reinforcement learning algorithm is used to find the optimal policy. The optimal policy is achieved when the cache hit rate is maximized. The obtained simulation results show that the reinforcement learning algorithm significantly reduces the long-term average cost compared to several baseline methods. Similar to [5] and [4] , the authors in [12] use a Q-learning algorithm that is based on finding an optimal policy that minimizes the expected long-term cost. The obtained simulation results show that the Q-learning algorithm reduces the average cost approximately by a factor of three compared to the threshold algorithm. The threshold algorithm commands the edge node to send a new update from the sensor whenever the value of AoI is AoI +1 when a specific physical quantity is requested. The common performance metric in [4] , [5] and [12] is the on-demand AoI.

3.2 Deep Reinforcement Learning approach

The problem in [13] and [14] is formulated as MDP and solved by using DRL. The goal in [14] is to minimize weighted average AoI and energy consumption. On the other hand, the goal in [13] is to minimize the cost of content replacement in the cache and the average AoI of the neural networks that are delivered to users. The authors in [13] assume a heterogeneous network with a macrobase station (ion (MBScontentontent servers (CSs and two neural networks (NNs. Each CS is responsible for one NN. The first NNNcontrols the navigation system, and the other one is accountable for the computer vision applications. In [14] , the system contains 3elements,s which are IoT devices (sensors), edge nodes, and data consumers (mobile users). The edge node is responsible for caching the sensor's content based on the user's request. In the papers discussed above, the authors proposed a DQN approach for training the data to solve the huge state space. Also, the authors allege that the outcome of applying this scheduling policy is better results in simulation.

Conversely, in references [15] , [16] , [17] , [18] , [19] , [20] , [21] ,[22] and [23] discussed minimizing the AoI and solving it by using DRL. In [15] , the authors model the problem as a bilevel Markov decision process; in contrast, in [16] , the authors used an episodic Mark decdecision process. And the MDP formulated in [17] as a finite-state finite-action MDP. In papers [18] , [21] , and [22] , authors consider unmanned aerial vehicle-assisted Internet of Things (IoT) networks. In [19] and [23] , the authors consider the internet of vehicular networks. In [19] , they consider vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications; on the other hand, in [23]

, they consider vehicle-to-everything (V2X) communications.

3.3 Cache Replacement Policies

The goal in [24] is to reduce the data traffic in the back-haul links by finding the optimal cache replacement policy that maximizes the cache hit ratio without knowing the popularity profile distribution, probability distribution of items, or user preference model. The system model contains end-user, cache-enabled base station services, and cloud services. The authors developed an online caching policy via Q-learning. The simulation result shows that the Q-learning-based approach significantly improved the cache hit ratio in comparison to the baseline methods.

The main goal in [25] is to solve the problem of caching IoT data at the edge without knowing future IoT data popularity, like user request patterns and other context characteristics, to balance the communication cost and the loss of data freshness. The MDP is modeled with two states, which are data freshness and edge-catching, which represent the values of information. The action in the MDP model is the decision of whether to cache a new data item when the data item arrives. The cost function will be a trade off between data freshness and communication cost when fetching IoT data.

3.4 Discussion

Figure 3.1 summarizes various Markov Decision Process (MDP) models utilized in different studies within the literature review. Each reference presents a unique approach

tailored to specific network environments and optimization objectives.

In [4], [5], and [12], the MDP models focus on Age of Information (AoI) and battery level considerations within IoT networks, aiming at either instantaneous AoI reduction or long-term cost minimization.

In contrast, [13], [14], and [15] extend their MDP models to heterogeneous networks, considering AoI reduction alongside factors such as energy consumption due to cache updates or content delivery choices.

Moreover, [16], [17], and [18] introduce MDP models tailored for specific IoT scenarios, such as information-centric IoT systems or UAV-assisted networks, targeting rewards related to AoI improvement while considering resource consumption factors.

Furthermore, [19], [20], and [23] explore MDP models in UAV-assisted networks, vehicular networks, and cellular vehicle-to-everything (C-V2X) communications networks, respectively, emphasizing AoI reduction and energy efficiency.

Lastly, [24] and [25] focus on cache-enabled base stations and data caching in wireless communication networks, with rewards centered around cache hit ratio optimization and communication cost reduction while maintaining data freshness.

These diverse MDP models reflect the complexities of modern communication networks and demonstrate tailored approaches to address specific challenges while optimizing various objectives, ranging from AoI reduction to energy efficiency and resource utilization.

Ref	State	Action	Rewards/ cost	Network
[4]	[AoI, Battery level]	[update the sensor or not]	Instantaneous AoI for each source	IoT Network
[11]	[AoI, Battery level]	[update the sensor or not]	Instantaneous AoI for each source	IoT Network
[12]	[AoI, Battery level]	[update the sensor or not]	minimizes the expected long-term cost	IoT Network
[13]	AoI , user requests	[stays idle, update the NNs]	A combination of the average AoI reduction and the cost associated with content updating	heterogeneous network
[14]	AoI , user request	[push the new version of content item, not]	average AoI reduction and the cost associated with energy consumption due to cache updates	IoT Network
[15]	AoI, status update sampling	Decisions related to offloading and scheduling policies, including choices about direct offloading or local computation and delivery, as well as scheduling decisions	average weighted sum of AoI reduction and the cost associated with energy consumption.	Multi device computing-enabled IoT system with a common destination.
[16]	[Age of Correlated Information (AoCI) , transmission status]	scheduling decisions of each slot	Improvement in AoCI is the primary reward, and the cost is associated with the resources expended in fetching or updating information.	Information-centric Internet of Things (IoT) system.
[17]	AoI values, energy states of source nodes	Decisions related to the online sampling policy, optimizing WPT, and scheduling of update packet transmissions.	Reduction in the long-term average weighted sum of AoI values is the primary reward, and the cost is associated with energy consumption, wireless energy transfer, or other resources expended.	wireless sensor network
[18]	virtual queues, AoI, UAV altitude, and channel state information	Decisions related to online scheduling, UAV altitude control, and management of virtual queues	Reduction in the (EWSA) is the primary reward, and the cost is associated with factors like energy consumption, reliability, and efficiency in relaying information	UAV-assisted Internet of Things (IoT) network
[19]	UAV location, sensor energy levels, age of information (AoI),	Decisions related to the UAV's flight trajectory, duration of data collection at each sensor, and transmission scheduling of the sensors.	Reduction in the weighted sum of (AoI) is the primary reward, and the cost is associated with factors like energy consumption, flight duration, and efficiency in data collection.	UAV-assisted Internet of Things (IoT) network
[20]	UAV and vehicle positions, age of information (AoI)	Decisions related to UAV trajectories, data collection and processing scheduling policies	Reduction in the (EWSA) is the primary reward, and the cost is associated with factors like trajectory complexity, scheduling policies, throughput constraints, and possibly energy consumption.	UAV-assisted single-hop vehicular network
[19]	current allocation of channels, transmission power levels, Age of Information (AoI)	Decisions related to spectrum and power dynamic allocation to minimize the sum of the average Age of Information (AoI) of all links and the average power consumption of all vehicular user pairs.	Reduction in the sum of the average (AoI) and average power consumption is the primary reward, and the cost is associated with factors like the complexity of the allocation scheme, interference, and efficiency of spectrum and power usage.	Internet of Vehicles (IoV)
[23]	vehicles' positions, directions, speeds, communication ranges, and transmission powers, the channel gains between vehicles, the current AoI of each pair of vehicles, and the current transmission period	Decisions related to transceiver selection, broadcast coverage optimization, power allocation, and RB scheduling to minimize the age of information (AoI).	Reduction in the (AoI) is the primary reward, and the cost is associated with factors like power consumption, coverage optimization, and the efficiency of resource block scheduling.	Cellular vehicle-to-everything (C-V2X) communications network
[24]	Information about the current status of the cache-enabled base station, contents of the cache, items requested by users, and the time-dependent nature of user preferences	Decisions related to cache replacement, determining whether to replace an item in the cache or discard a fetched item	Improvement in the cache hit ratio is the primary reward, and the cost is associated with the efficiency of cache replacement decisions and potential data traffic on back-haul links	wireless communication network
[25]	cache contents, data freshness, IoT data popularity, user request patterns,	decisions related to caching IoT data at the network edge, including which data to cache, when to evict or replace data, and how to balance communication cost and data freshness.	Optimization of the balance between communication cost and the loss of data freshness is the primary reward, and the cost is associated with factors like communication resources, data freshness, and the efficiency of caching policies.	Wireless communication network

Figure 3.1: Comparison between the MDP models in the references.

CHAPTER 4

SYSTEM MODELING

4.1 System model

We abstract the system from Figure 1.1 as shown in Figure 4.1 . In this system, The cache contains all the states of the sources. The sensors will read the states of the sources and update the cache. Since the relationship is many-to-many, the states of the sources can be captured by multiple sensors. Furthermore, the application in the cloud can request the states of more than one source. at the same time. The three layers in Figure 1.1 have the following responsibilities:

1. Cloud layer: The IoT application runs in the cloud; it could be multiple applications. The application sends a request(s) to the edge node for a particular source. The request contains the source IDs that the age-sensitive application monitors.
2. Edge layer: All IoT device updates are collected in this layer and cached. The newly A DRL-based algorithm will be implemented in this layer.

3. IoT Infrastructure layer: This layer is composed of sensors and sources. The sensors are responsible for capturing the source data and updating the cache on the edge. The sources could be a physical phenomenon or autonomous cars.

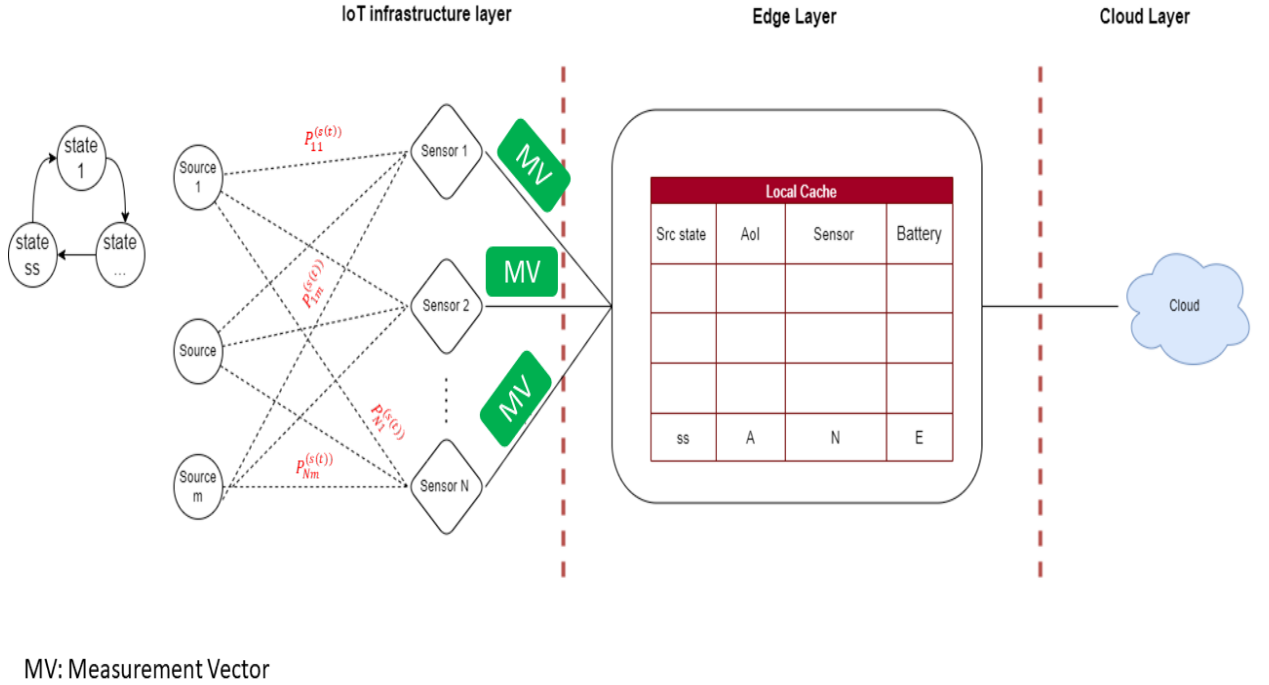


Figure 4.1: System model.

4.2 MDP model

We were inspired by the work in [1] to generalize the model in Table. 4.1 compares our newly developed model with the models in [4] and [1]. In the model in [4], the application will target a specific sensor to get the update. On the other hand,

in our model, the relationship between the sources and the sensor is many-to-many, which leads to a configuration in which any source can be captured by any sensor. Furthermore, in our model, more relevant state variables are included (see equation 4.1), which are the source state SS , AoI A , battery E , N is the number of sensors, and M is the number of sources. These new state variables will add more dimensions to the original state space proposed in [4]. The resulting multi-dimensional state space will have an extremely large state space, as shown in Eq. 4.2. Where SS is the source state, A_{max} is the maximum AoI value, M is the number of the sources, E_{max} is the maximum battery level, and N is the number of sensors. As a result, we will solve this problem by using a DRL algorithm. For example, if the sensors 2 and 3 sources and 2 sources state the total system space will be around 512000, and if the sensors 2 and 2 sources and 2 sources state the total system space will be around 256000 if the AoI is 10 and the maximum battery level is 7.

We generalize the model in [4] because:

- The model in [4] considers the on-demand AoI. Other sources that we review just consider the AoI.
- In the paper [14], they assume an IoT network, but they didn't consider the battery or energy consumption.
- On the other hand, in our model, we consider both battery and energy consumption. In the literature review, consider other types of networks, such as heterogeneous networks.
- Our cost function includes the client's requests. Other sources that we reviewed

	[4]	[1]	Proposed model
SV	[AoI, Battery level]	[source state, AoI]	[(SS, AoI), (S, B)]
A	[update the sensor or not]	finite action space 1,..., N	Update (S1, ..., Sn)
C	Instantaneous AoI for each source	Average AoI for all sources	Ave on-demand AoI

Table 4.1: Generalized MDP models and proposed MDP model.

did not consider the user request on the cost function. Figure 3.1 compares our MDP model with MDP models in the literature review.

For example, in references [13], we can generalize and use the on-demand AoI as a performance matrix, but the authors consider a different type of network, which is a heterogeneous network.

$$State = (SS, A)^M \cup (E)^N \quad (4.1)$$

$$\text{Total Number of System State} = (SS \times A_{\max})^M \times (E_{\max} + 1)^N \quad (4.2)$$

For the action, there is one set of actions in our model, which is which sensor should be selected to update the source state. See equation 4.3 , where N is the the number of sensors. This action set will provide a high quality of the source state and its age (age of information).

$$Actions = \{1, 2, \dots, N\} \quad (4.3)$$

The state transition probability matrix that captures the dynamic of the system is not included in our model since we consider it a mode- free RL. Therefore, we only identify the state, action, and cost in our model. The coverage matrix represents the relationship between the sensors, sources, and source states. Tables 4.2 , 4.3 , and 4.4 show the convergence matrix of environment cases 1, 2, and 3 and the measurement vector, which contains the current state of the source if it is covered by the sensor. The coverage matrix will be implemented in the IoT Infrastructure layer, and it will return 0 if the sensors, sources, and source state are not covered; otherwise, it will return 1. In this model, we categorized two types of coverage matrices. The first is the fully covered environment, followed by the half-covered environment.

1. Fully Covered Environment: for the fully covered environment, all sensors, sources, and sources states are covered in this case, which will ensure that the transmission will always be successful. Figures 4.2, 4.3 , and 4.4 show the fully covered environment where all links between the sensors and sources were connected.
2. Half-Covered Environment: for the half-covered environment, half of the sen-

sors, sources, and sources states are covered in this case, which means that the transmission will not always be successful. We assume that in the half-covered environment, the links in Figures 4.2 , 4.3 , and 4.4 are not all connected.

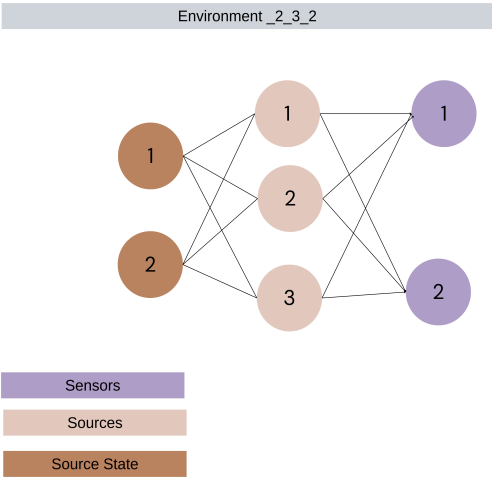


Figure 4.2: Fully Covered Environment for case 1.

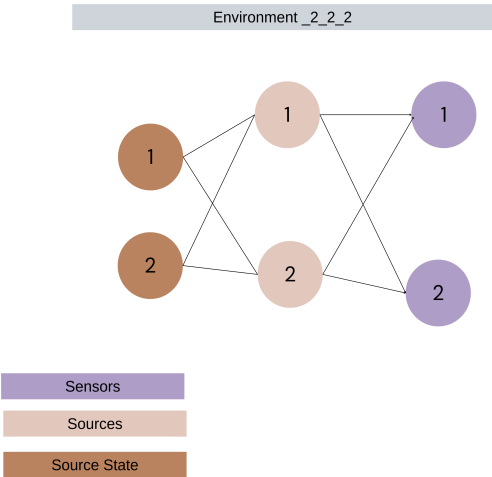


Figure 4.3: Fully Covered Environment for case 2.

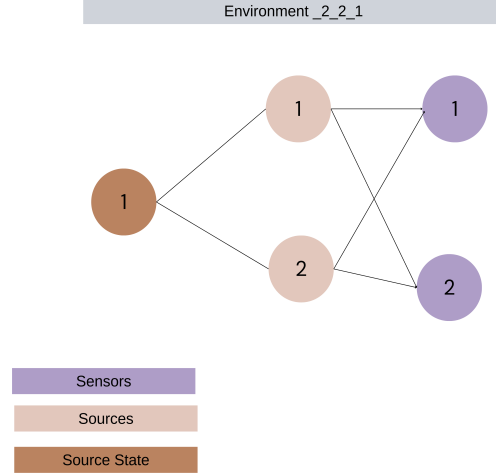


Figure 4.4: Fully Covered Environment for case 3.

The cost function depends on the average weighted on-demand AoI. That is requested from the client. This cost function will capture the total number of clients demand for each source state in the environment multiplied by the age of the current state of the system. R_{src} is the number of requests for the selected source, whereas R_{Total} is the total number of generated requests, as shown in equation 4.4 .

$$C_{total} = \sum_{i=1}^M \left(\frac{R_{src}}{R_{Total}} \right)_i * (\Delta(t+1))_i \quad (4.4)$$

	Sensor	Sources	Source State
1	0	0	0
2	0	1	0
3	0	2	0
4	0	0	1
5	0	1	1
6	0	2	1
7	1	0	0
8	1	1	0
9	1	2	0
10	1	0	1
11	1	1	1
12	1	2	1

Table 4.2: Coverage Matrix of environment case 1 .

	Sensor	Sources	Source State
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Table 4.3: Coverage Matrix of environment case 2.

	Sensor	Sources	Source State
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	0

Table 4.4: Coverage Matrix of environment case 3.

CHAPTER 5

SIMULATION

5.1 Environment

The environment is implemented by using a Python class. We consider in our environment that we have 2 sensors, 3 sources, and 2 source states. See figure 5.1 . We categorized the class into functions, which are:

- Initialize function: in this function, we initialized all needed parameters, such as the state space, action space, coverage matrix, source MDP, state vector, etc.
- Initialize Coverage Matrix Function: This function displays the connectivity between the sensors and sources. We can modify this function to make the environment a fully covered environment or a half-covered environment.
- Initialize Source Markov Process: initialize the probability between the sources.
- Update Battery Level: in this function, the battery level of the sensors will be updated.

- Reset function: this function will reset all values, like state and battery level.
- Step function: this is the major function in the environment, which contains multiple steps:
 1. Construct the measurement vector (MV) that will be returned by the sensor (i.e., action).
 2. Transmit the measurement vector.
 3. Compute Cost by Generating the Request Vector (RV) and Compute the Average On-Demand AoI.
 4. Generate next states for sources.
 5. Harvest energy to re-charge batteries.
 6. Return to the next state and the cost.

Moreover, in this paper, we consider two more environments:

- Two sensors, two sources, and two source states; see figure 5.2 .
- Two sensors, two sources, and one source state; see figure 5.3 .

To study the different behaviors of the environments, which will be discussed in the next chapter.

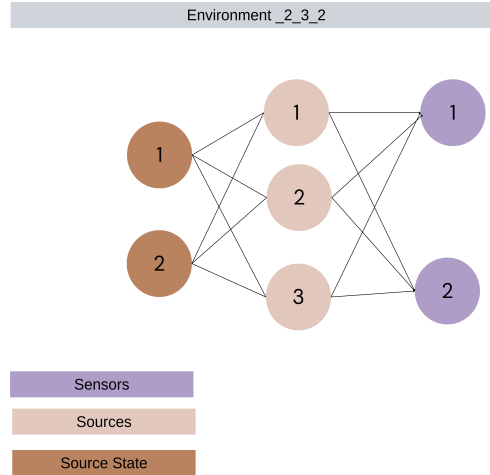


Figure 5.1: Environments case 1.

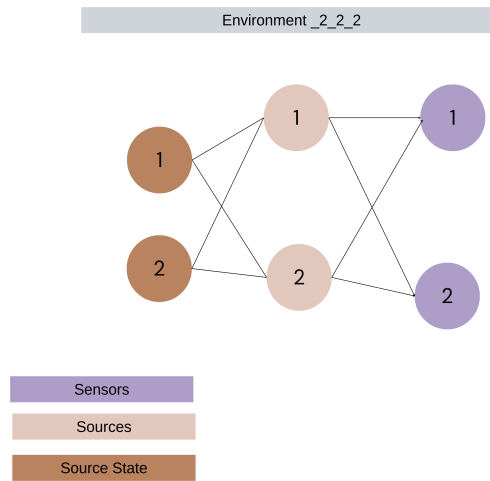


Figure 5.2: Environments case 2.

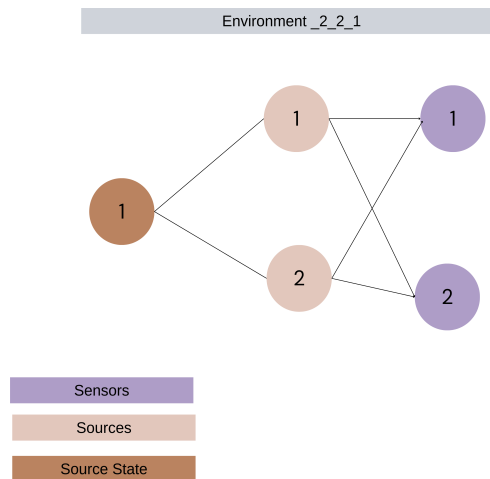


Figure 5.3: Environments case 3.

The first step in the simulation is checking the coverage matrix; if it is covered, the measurement vector will be created. After that, the battery level will be checked to see if it is capable of transmitting the data; the probability of a successful transmission will be checked; otherwise, the source state will not be updated, and the AoI value will be incremented. After checking the probability of a successful transmission, if the transmission was successful, the source state will be updated as well as the AoI value. On the other hand, if the transmission was not successful, the source state will not be updated and the AoI value will be incremented (see figure 5.4). After updating the source state and AoI values, the cost will be calculated as well as the next state. then it will be returned to the agent (Q-learning or DqN agent).

The system parameters were selected based on the literature review and the problem itself. Since the main goal is to minimize the AoI, we have to make sure that the transmission is successful. This leads to a high probability of a successful transmission.

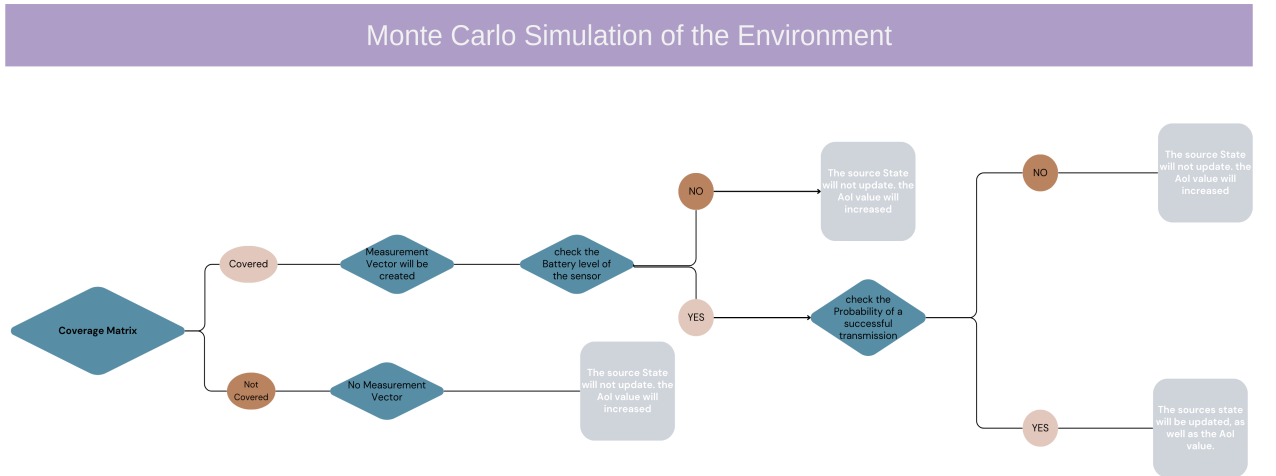


Figure 5.4: Flowchart for the simulation of the environment.

5.2 Q-Learning Algorithm

The Q-learning algorithm uses the q-table to achieve the optimal policy for our environment. Algorithm 1 shows the q-learning algorithm that is used in this study. First, we initialize all the needed parameters. Then, we start with two loops for episodes and epochs. In each episode, we will call the reset function from the environment to get the initial state, which is shown in line 5. In each epoch, we check if the randomly generated number is less than the epsilon line 7, and a random action will be selected from the environment action space line 8. Otherwise, the action will be selected based on the minimum action from the q-table. In line 12, the action will be applied to the environment, and then the cost and next state will be observed. In line 14, the q-value will be updated.

Algorithm 1 Q-learning Algorithm

```
1: Initialize Q-table  $Q$ 
2: Set learning rate  $\alpha$ , discount factor  $\gamma$ , exploration probability  $\epsilon$ 
3: Initialize state  $s$ 
4: for episode = 1 to  $i$  do
5:   Reset environment to initial state
6:   for Epochs = 1 to  $j$  do
7:     if random number  $\leq$  epsilon then
8:       Choose a random action from action space in the environment.
9:     else
10:      Choose a Minimum action from Q-table
11:    end if
12:    apply the action in the step function in the environment.
13:    observe cost  $c$  and the new state  $s'$ 
14:    Update Q-value:  $Q_m(s_m(t), a_m(t)) \leftarrow (1 - \alpha(t))Q_m(s_m(t), a_m(t)) + \alpha(t)(c_m(t) + \gamma \min_{a \in A_m} Q_m(s_m(t+1), a))$ 
15:    Set  $s \leftarrow s'$ 
16:  end for
17: end for
```

5.3 DqN Algorithm

Deep Q Networks is a model-free reinforcement learning algorithm. Rather than the classic Q-learning algorithm, DqN has three new concepts, which are using the deep convolutional neural network architecture for approximating the Q-value, replacing single-step updates based on the previous experience with mini-batches of random training data, and calculating the Q-values of the future state with the help of existing network parameters [26] . DqN uses exploration and exploitation strategies to find the optimal policy. First of all, during the training, the DqN algorithm will explore more actions and different states to collect more diverse experiences. Then, after exploring the face, the DqN agent will exploit the learned knowledge from the training to make more optimal decisions based on the predicted Q-values.

To complete the training face in DqN, DqN Agent needs two neural networks, which are the online network and the target network. For the online network, it uses the replay memory to sample a mini-batch of states,actions, rewards, and next states for observing the Q-Network(target network). After sampling the mini batch, the DqN agent will approximate the action-value function. Target networks are used for predicting the action in a given state. The target network is simply a copy of the main network (the online network), but with a lower updating rate to stabilize the learning process of DqN [27] .

Algorithm 2 explains the DqN algorithm used in this paper. In the beginning, we initialize all the needed parameters, which are:

- Target Q-network with weights

- Replay memory
- Exploration rate
- Batch size
- Discount factor
- Number of episodes and Epochs

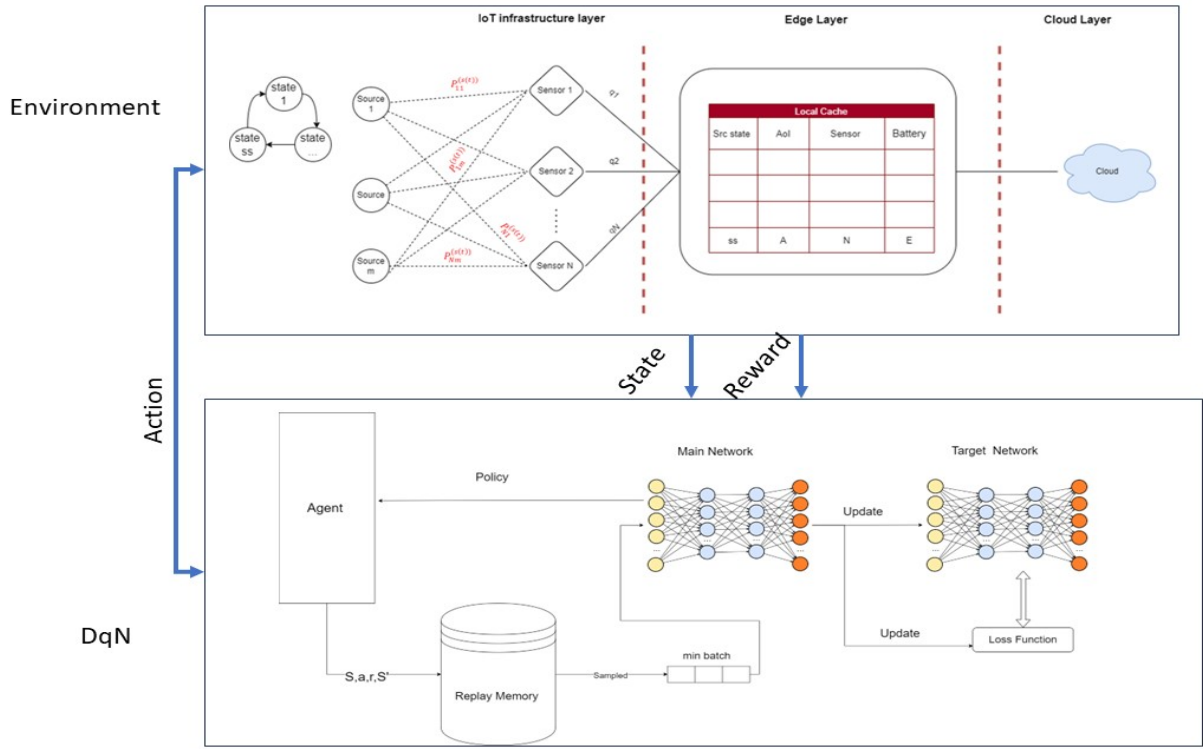


Figure 5.5: DQN diagram.

Starting from line 15 of algorithm 2, the exploring face in DQN starts. Line 17 shows the exploited face. Then, the action will be applied to the environment, and the cost will be received. Lines 15 to 18 show the sampling (the online network). Lines 19 and 20 are the experience replay (replay memory) steps.

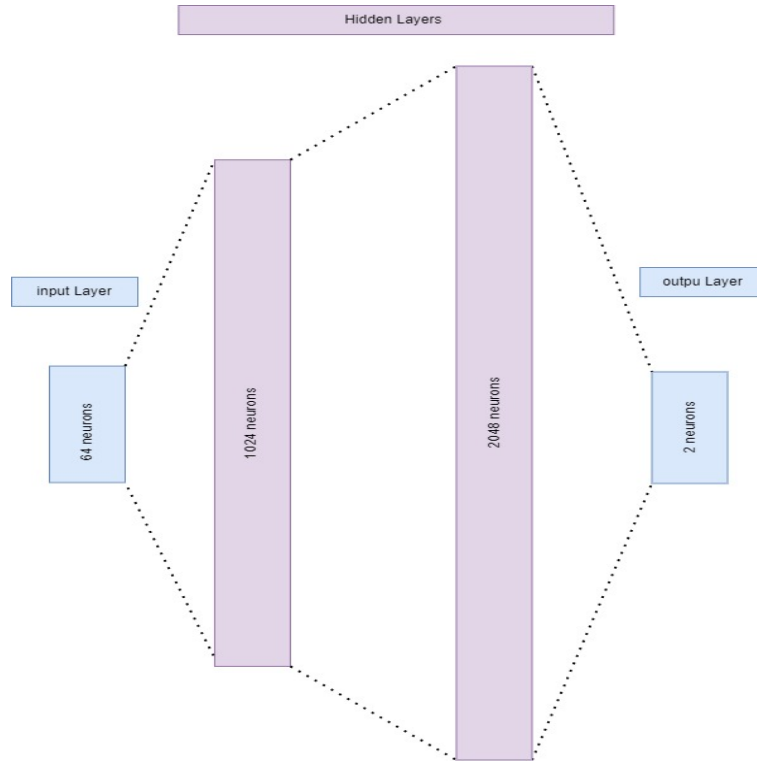


Figure 5.6: DRL architecture.

The total DNN layers are 4 layers, of which 2 are hidden layers. The first hidden layer contains 1024 neurons, and the second hidden layer contains 2048 neurons. see figure 5.6 The activations used in the DNN are relu and linear with the Adam optimizer.

Algorithm 2 Deep Q-Network (DQN) Algorithm.

Environment E , Replay memory D , Q-network parameters θ
2: Optimal Q-network parameters θ^*
 Initialize Q-network with random weights θ
4: Initialize target Q-network with weights $\theta^- = \theta$
 Initialize replay memory D
6: Initialize exploration rate ϵ
 Initialize batch size B
8: Initialize discount factor γ
 Initialize maximum number of episodes N
10: Initialize maximum number of Epochs per episode T

12: **for** *episode* e **do**
 Initialize state s_1
14: **for** $t = 1$ T **do**
 if random() $\leq$ probability ϵ **then**
16: select a random action a_t
 else select $a_t = \arg \max_a Q(s_t, a; \theta)$
18: Execute action a_t , observe reward r_t and next state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in D
20: Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
 Set target $y_j = r_j + \gamma \min_{a'} Q(s_{j+1}, a'; \theta^-)$
22: Update Q-network
 Update target Q-network weights: $\theta^- = \tau \theta + (1 - \tau) \theta^-$
24: **if** s_{t+1} is terminal **then** Break
26: **if** $\epsilon > \epsilon_{\min}$ **then** Decrease ϵ by a small amount

5.4 Greedy Algorithm

We consider the greedy algorithm based on the sensor's coverage. In this greedy algorithm, we select the sensor to update the source state if it covers more sources. As shown in algorithm 3, $C_{sourcesn}$ is the total number of covered sources for the sensor, and n represents the sensor number. If the first sensor covers more sources than sensor 2, sensor 1 will be selected for the update. Otherwise, the second sensor will be selected for the update. If both sensors cover an equal number of sources, a random action will be selected to update. In line 13, the cost and next state will be observed.

Algorithm 3 Greedy Algorithm for coverage for minimizing AoI.

```

Initialize  $a \in A_m, m \in M, n \in N, b \in B, C_{sensorn}, C_{sourcesm}$  .
for Episodes in  $e$  do
    Initialize state  $s_1$ 
4:   for Epochs in  $p$  do
        if  $C_{sourcesn1} > C_{sourcesn2}$  then
            Action =  $N_{sensorn1}$ 
        else if  $C_{sourcesn1} < C_{sourcesn2}$  then
8:         Action =  $N_{sensorn2}$ 
        else
            Action = random choose
        end if
12:    Apply the action in the environment and observe the next state and the
        cost.
    end for
end for=0

```

5.5 Random Algorithm

In the random algorithm, we select a random action from the action space in our environment, and then we apply the randomly selected action to the environment.

Then, we capture the cost each time we apply the randomly selected action to see the performance of the random algorithm. See Algorithm 4 .

Algorithm 4 Randomly action algorithm .

Initialize $a \in A_m, m \in M, n \in N, b \in B, C_{sensorn}, C_{sourcesm}$.

for Episodes in e **do**

 Action = random choose

 Apply the action in the environment and observe the next state and the cost.

5: **end for**

CHAPTER 6

SYSTEM EVALUATION

6.1 System Evaluation Over Time

In this section, we will evaluate different environments for the fully covered environment and the half-covered environment in environment cases 1, 2, and 3.

6.1.1 Environment Case 1

In this environment, we assume that we have 2 sensors, 3 sources, and 2 source states.

The environment parameters are set as shown in the table. 6.1 .

Parameter	Value
Number of clients	5
Maximum battery size	7
Maximum age of information	10
Probability of a successful transmission	1.0
Probability that a client requests a source	0.3
Probability that a sensor harvests one unit of energy	0.2

Table 6.1: System Parameters.

Fully Covered

Figure 6.1 illustrates that the DqN algorithm shows a noticeable minimization in the average on-demand AoI compared to other algorithms. Moreover, the random algorithm reaches the highest value of the average- on demand AoI. After the random algorithm, the greedy algorithm takes second place in given a high value of the average on demand AoI. Following that, Q-Learning becomes better performance compared to random and greedy algorithms.

Half Covered

Also, in the half-covered environment, which is shown in figure 6.2 , the performance of the algorithms shows that the DqN is improved in minimizing the average on-demand AoI compared to Q-Learning, Greedy, and Randomly selected algorithms.

In this environment, the performance of Q-Learning in the middle stage comes closer to DqN but still has a higher average on demand AoI. And the cause of this behavior is the drop in coverage in the environment since we assume that source state number 2 is not covered in this scenario.

Table 6.2 shows the average on deemed AoI for fully covered and half covered. As shown, DqN in the half-covered environment increased by 1.06 % from the fully covered environment. The half-covered Q-Learning increases by 3.48 % compared to the fully covered environment. Moreover, the average AoI in the DqN algorithm is minimized in the fully covered environment by 10.3% compared to Q-Learning, a decrease of 18.31% compared to the Greedy algorithm, and 30.95% compared to the Random algorithm in the 2 3 2 environment. Additionally, the average AoI in the

DqN algorithm is minimized in the half-covered environment by 13.39% compared to Q-Learning, a decrease of 17.7% compared to the greedy algorithm, and 29.5 % compared to the random algorithm. See table 6.3 .

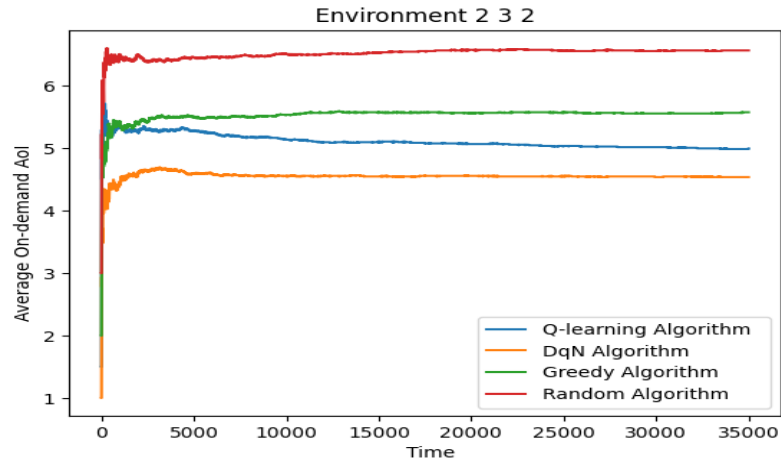


Figure 6.1: environment case 1 Fully Covered

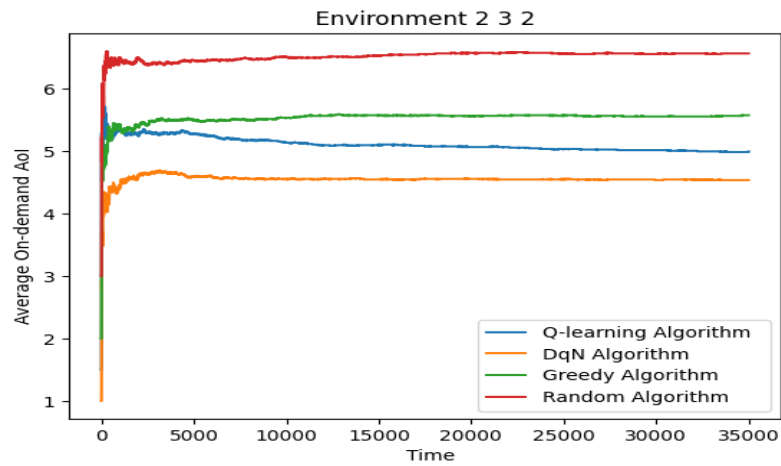


Figure 6.2: environment case 1 half Covered

Algorithm	Average AoI Fully covered	Average AoI Half covered
DqN Algorithm	4.537	4.585
Q-Learning Algorithm	5.058	5.234
Greedy Algorithm	5.554	5.575
Random Algorithm	6.571	6.505

Table 6.2: Average on demand AoI of fully covered and Half covered environment case 1

DqN		
	Fully Covered	Half Covered
Q-Learning Algorithm	10.3%	13.39 %
Greedy Algorithm	18.3%	17.7%
Random Algorithm	30.95%	29.5%

Table 6.3: Percentage of minimizing for fully covered and Half covered environment case 1

6.1.2 Environment Case 2

In this environment, we assume that we have 2 sensors, 2 sources, and 2 source states.

The environment parameters are set as shown in the table. 6.1 .

Figure. 6.3 shows the behavior of the system in a fully covered environment. On the other hand, Figure 6.4 shows the half-covered environment.

For both environment-covered and half-covered environments, the DqN algorithm shows a noticeable minimizing on the average on-demand AoI. Comparing to Q-Learning in full coverage, the amount of decreasing is 9.98%, 35.85% to the greedy algorithm, and 41.76% to the random algorithm. the percentages of decreasing in the DqN algorithm of the half-covered environment, 15.64% compared to the Q-Learning algorithm, 21.97% compared to the greedy algorithm, and 26% compared to the random algorithm. See tables 6.4 and 6.5 .

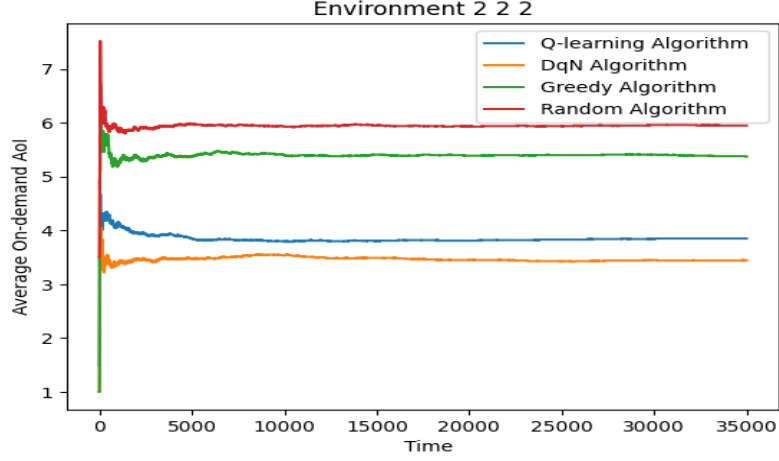


Figure 6.3: environment case 2 fully Covered

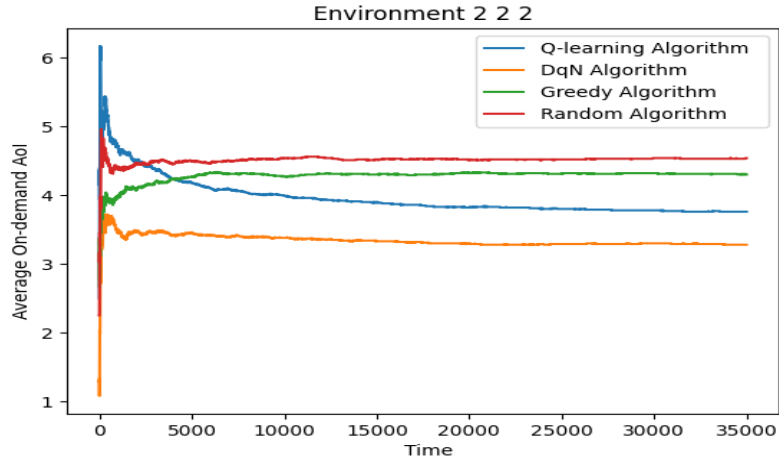


Figure 6.4: environment case 2 half Covered

6.1.3 Environment Case 3

In this environment, we assume that we have 2 sensors, 2 sources, and 1 source state.

The environment parameters are set as shown in the table. 6.1 .

Figures 6.5 and 6.6 demonstrate environment case 3 for covered and half covered. Similarly to environments 1 and 2, in environment 3, the DqN algorithm has the best performance compared to the Q-Learning algorithm, the greedy algorithm, and the random algorithm. The average on-demand AoI in DqN is minimized by 13.66% com-

Algorithm	Average AoI Fully covered	Average AoI Half covered
DqN Algorithm	3.462	3.339
Q-Learning Algorithm	3.846	3.958
Greedy Algorithm	5.397	4.279
Random Algorithm	5.944	4.512

Table 6.4: Average on demand AoI of fully covered and Half covered environment case 2

DqN		
	Fully Covered	Half Covered
Q-Learning Algorithm	9.9%	15.64 %
Greedy Algorithm	35.8%	21.97%
Random Algorithm	41.7%	26%

Table 6.5: Percentage of minimizing for fully covered and Half covered environment case 2

pared to Q-Learning, 36.01% compared to the greedy algorithm, and 37.36% compared to the random algorithm in the fully covered environment. The average on-demand AoI in DqN is minimized by 15.66% compared to Q-Learning, 23.15% compared to the greedy algorithm, and 31.59% compared to the random algorithm in the half-covered environment. See tables 6.6 and 6.7 .

Overall, the DqN algorithm shows a positive performance in minimizing average on-demand AoI around all environments for fully covered and half-covered environments.

Algorithm	Average AoI Fully covered	Average AoI Half covered
DqN Algorithm	3.412	3.286
Q-Learning Algorithm	3.952	3.880
Greedy Algorithm	5.332	4.276
Random Algorithm	5.447	4.804

Table 6.6: Average on demand AoI of fully covered and Half covered environment case 3

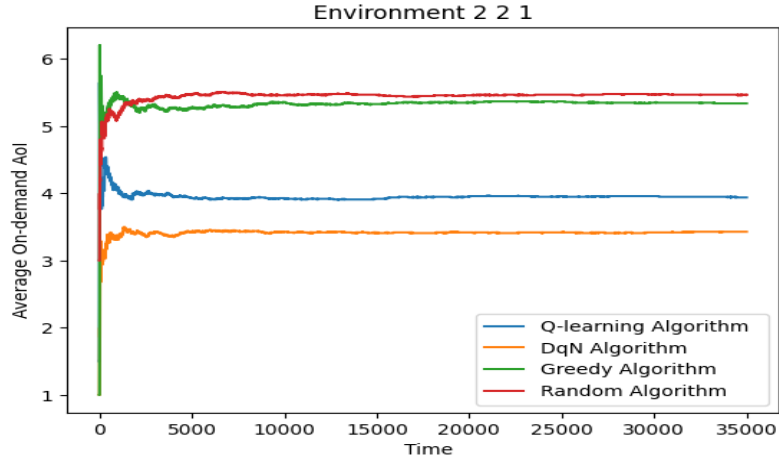


Figure 6.5: environment case 3 fully Covered

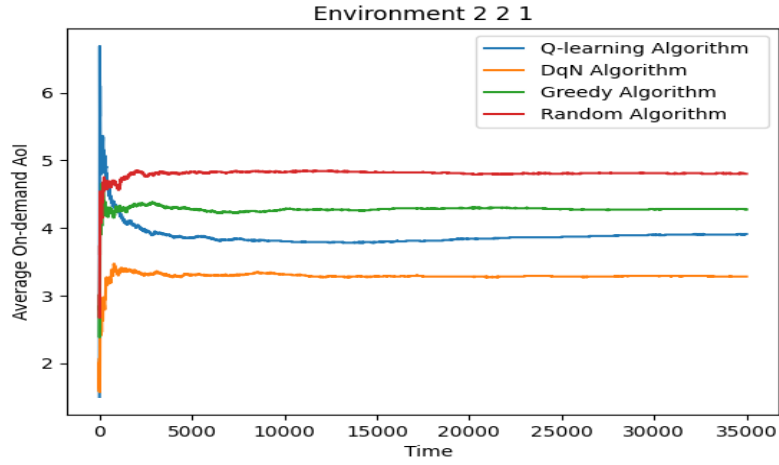


Figure 6.6: environment case 3 half Covered

DqN		
	Fully Covered	Half Covered
Q-Learning Algorithm	13.66%	15.30%
Greedy Algorithm	36.01%	23.15%
Random Algorithm	37.3%	31.59%

Table 6.7: Percentage of minimizing for fully covered and Half covered environment case 3

6.2 System Evaluation Under Changing Environment Parameters

In this section, we will evaluate the system behavior while changing the environmental parameters, such as the probability of a successful transmission and the probability that a sensor harvests one unit of energy. We used the optimal policies to gather the system behavior. In the DqN case, we upload the trained model, and for Q-Learning, we upload the Q-table.

6.2.1 Changing Probability of a successful transmission

Figure. 6.7 illustrates the system behavior while changing the probability of a successful transmission in the DqN algorithm. As the probability increases, the cost keeps decreasing. The maximum cost was around 9.5, and the minimum was around 4.

Conversely, Q-learning behavior while changing the probability of a successful transmission shows a negative correlation, which means that when the probability increases, the cost keeps decreasing. See Figure. 6.8 .

Figures 6.9 and 6.10 illustrate the system behavior while changing the probability of a successful transmission in the greedy algorithm and the random algorithm, respectively. The greedy algorithm reached a minimum of 5.6 in average cost, while the random algorithm reached 6.5. Both the greedy algorithm and the random algorithm show a slight reduction in the average on-demand AoI, which leads to the conclusion

that DqN and Q-learning have better performance in this study.

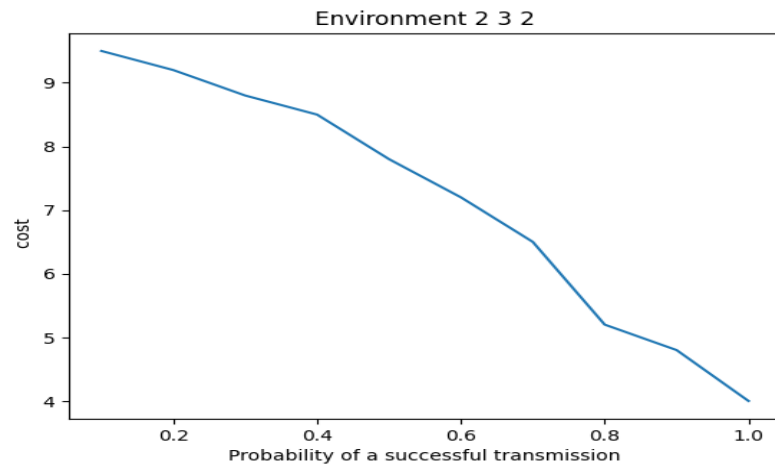


Figure 6.7: environment case 1 Changing Probability of a successful transmission DqN

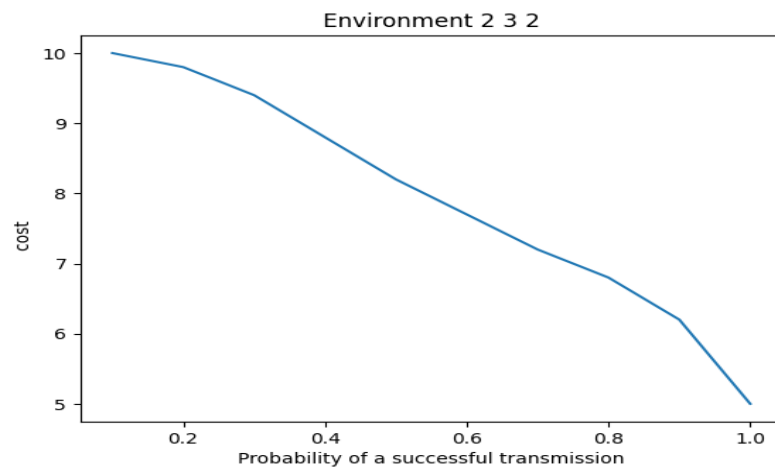


Figure 6.8: environment case 1 Changing Probability of a successful transmission Q-Learning

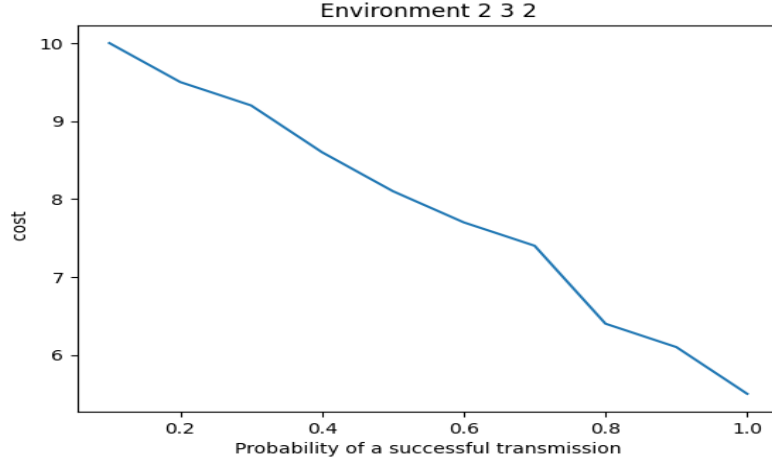


Figure 6.9: environment case 1 Changing Probability of a successful transmission Greedy

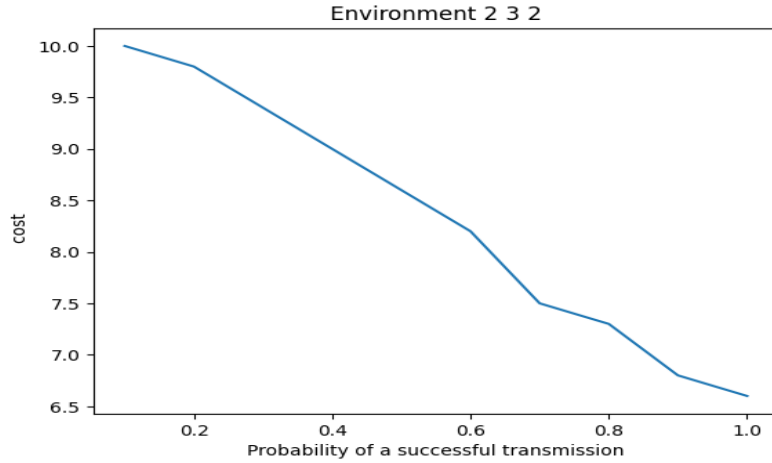


Figure 6.10: environment case 1 Changing Probability of a successful transmission Random

6.2.2 Changing Probability of Energy Harvesting

Figure 6.11 shows the DqN behavior while changing the probability that a sensor harvests one unit of energy, and Figure 6.12 shows the Q-Learning behavior. Both Q-Learning and DqN are sharing the almost same behavior, which is that the cost values are decreasing when the probability is increasing.

Figures 6.13 and 6.14 illustrate the system behavior while changing the proba-

bility of a successful transmission in the greedy algorithm the random algorithm, respectively. Greedy reached a minimum of 6.5 in average cost, while the random algorithm reached 7.0. Both Greedy and Random show a slight minimizing in the average on-demand AoI, which leads to the conclusion that DqN and Q-learning have better performance in this study.

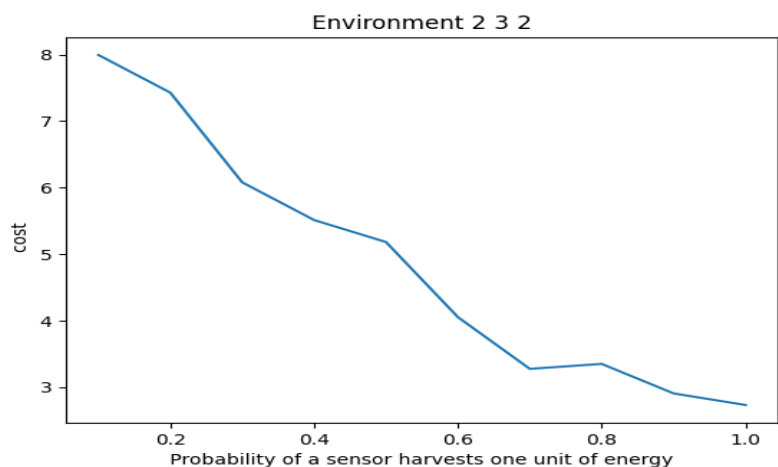


Figure 6.11: environment case 1 Changing Probability that a sensor harvests one unit of energy DqN

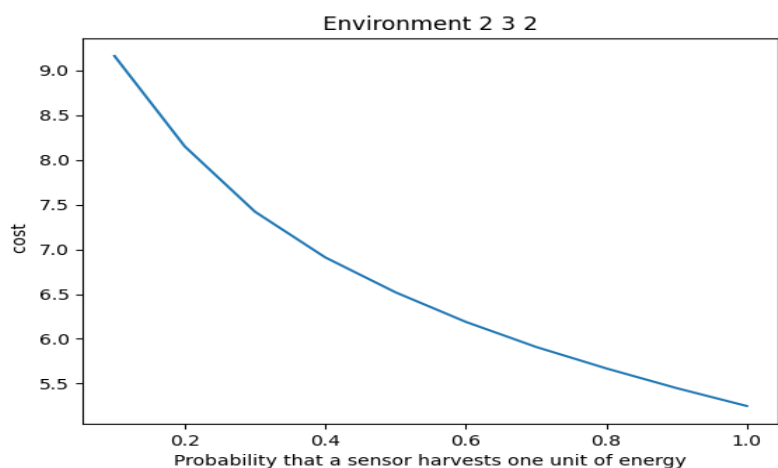


Figure 6.12: environment case 1 Changing Probability that a sensor harvests one unit of energy Q-Learning

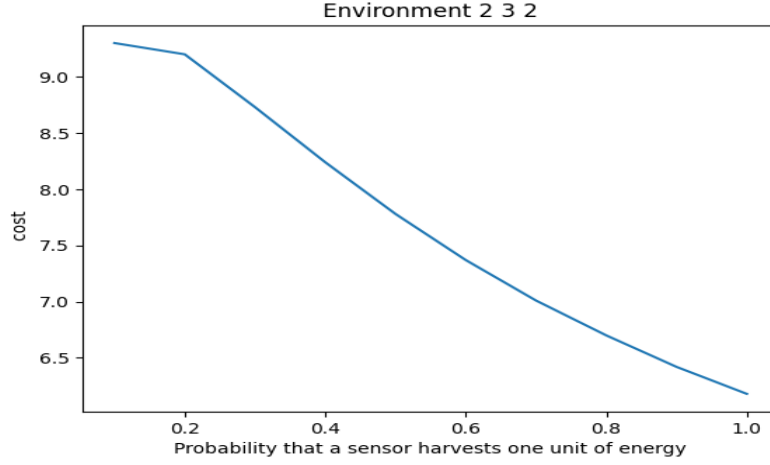


Figure 6.13: environment case 1 Changing Probability that a sensor harvests one unit of energy Greedy

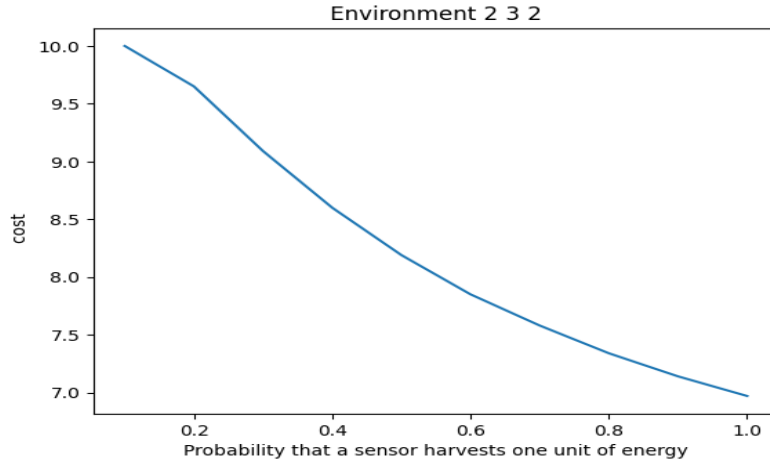


Figure 6.14: environment case 1 Changing Probability that a sensor harvests one unit of energy Random

6.3 Discussion

After going through all different scenarios and environments, the DqN algorithm shows better performance in all environments in minimizing the average on-demand AoI. Then, the Q-learning algorithm, the greedy algorithm and lastly, the random algorithm. We can summarize the evaluation at these points:

- DqN Algorithm is minimizes of all environments in the fully covered case by

11.2 % compared to the Q-Learning algorithm, 30% compared to the greedy algorithm, and 36% compared to the random algorithm.

- DqN Algorithm minimizes average to all environments in the half-covered case by 14.44% compared to the Q-Learning algorithm, 20.94% compared to the greedy algorithm, and 29.03% compared to the random algorithm.
- Changing the environment parameters can lead to changes in the system's behavior.
- DqN shows a noticeable minimization in average on-demand AoI across all environments and scenarios.

CHAPTER 7

CONCLUSION

7.1 Conclusion

In conclusion, in this thesis, we have reviewed the relevant literature about Age of Information, Reinforcement Learning, and Deep Reinforcement Learning. Also, we have formulated a new research problem for minimizing on-demand AoI for age-sensitive IoT applications. This new research problem is a generalization of an existing problem in the literature.

Also, we have considered many to many relationships between sensors and sources. The simulation results show that the DRL algorithm achieved a reduction of 11.2% compared with the Q-Learning algorithm. Also, the DRL algorithm can achieve a reduction of 30% and 36% compared to the greedy algorithm and the random algorithm, respectively. Moreover, the DQN algorithm minimizes an average of all environments in the half-covered case by 14.44% compared to the Q-Learning algorithm, 20.94% compared to the greedy algorithm, and 29.03% compared to the random algorithm.

7.2 Future Work

Our objective function captures only the immediate average on-demand AoI. Another component can be added to the objective function, such as energy consumption and the fairness of sampling the sensors. Also, the cost of switching between sensors can be considered. Switching between sensors introduces new bipartite graphs that capture the many-to-many relationships between sensors and sources. Capturing the cost of switching will enable the RL agent to find more effective policies for operating the proposed system.

REFERENCES

- [1] A. E. Kalor and P. Popovski, “Timely monitoring of dynamic sources with observations from multiple wireless sensors,” *IEEE/ACM Transactions on Networking*, 2022.
- [2] mariamsuboh. (2022) Neom’s the line: The world’s first cognitive city. [Online]. Available: <https://wired.me/technology/neoms-the-line-the-worlds-first-cognitive-city/>
- [3] J. Machin and A. Solanas, “A review on the meaning of cognitive cities,” in *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE, 2018, pp. 1–5.
- [4] M. Hatami, M. Leinonen, and M. Codreanu, “Aoi minimization in status update control with energy harvesting sensors,” *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 8335–8351, 2021.
- [5] —, “Minimizing average on-demand aoi in an iot network with energy harvesting sensors,” in *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2021, pp. 1–5.

- [6] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, “Age of information: An introduction and survey,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.
- [7] A. Kosta, N. Pappas, V. Angelakis *et al.*, “Age of information: A new concept, metric, and tool,” *Foundations and Trends® in Networking*, vol. 12, no. 3, pp. 162–259, 2017.
- [8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [9] Y. Li, “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*, 2017.
- [10] S. S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: an overview,” in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*. Springer, 2018, pp. 426–440.
- [11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [12] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, “Age-aware status update control for energy harvesting iot sensors via reinforcement learning,” in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2020, pp. 1–6.

- [13] Y. Tang, G. Ma, H. Xie, J. Xu, and X. Han, “Joint transmit and reflective beamforming design for irs-assisted multiuser miso swipt systems,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.
- [14] X. Wu, X. Li, J. Li, P. Ching, and H. V. Poor, “Deep reinforcement learning for iot networks: Age of information and energy cost tradeoff,” in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.
- [15] X. Xie, H. Wang, and M. Weng, “A reinforcement learning approach for optimizing the age-of-computing-enabled iot,” *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2778–2786, 2021.
- [16] B. Yin, S. Zhang, and Y. Cheng, “Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach,” *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8748–8759, 2020.
- [17] M. A. Abd-Elmagid, H. S. Dhillon, and N. Pappas, “A reinforcement learning framework for optimizing age of information in rf-powered communication systems,” *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4747–4760, 2020.
- [18] M. Samir, C. Assi, S. Sharafeddine, and A. Ghrayeb, “Online altitude control and scheduling policy for minimizing aoi in uav-assisted iot wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2493–2505, 2020.

- [19] N. Peng, Y. Lin, Y. Zhang, and J. Li, “Aoi-aware joint spectrum and power allocation for internet of vehicles: A trust region policy optimization-based approach,” *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19 916–19 927, 2022.
- [20] Y. Lu, J. Gong, and X. Chen, “Joint aoi and latency optimization in iot systems based on deep reinforcement learning,” in *2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. IEEE, 2022, pp. 281–286.
- [21] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, “Deep reinforcement learning for fresh data collection in uav-assisted iot networks,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 716–721.
- [22] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghrayeb, “Age of information aware trajectory planning of uavs in intelligent transportation systems: A deep learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 382–12 395, 2020.
- [23] Z. Mlika and S. Cherkaoui, “Deep deterministic policy gradient to minimize the age of information in cellular v2x communications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23 597–23 612, 2022.
- [24] M. Hatami, M. Leinonen, and M. Codreanu, “Online caching policy with user preferences and time-dependent requests: A reinforcement learning approach,”

- in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 1384–1388.
- [25] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, “Caching transient data for internet of things: A deep reinforcement learning approach,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2074–2083, 2018.
- [26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [27] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A theoretical analysis of deep q-learning,” in *Learning for dynamics and control*. PMLR, 2020, pp. 486–489.

VITAE

- Name: Ala Mohammed Alhamdan
- Nationality: Saudi
- Date of Birth: 21.07.1995
- Email: *alaaalhamdan1995@gmail.com*
- Permenant Address: Saudi Arabia, Dhahran 34255-4747

EDUCATION :

- Bachelor's degree in Computer Engineering ,College of Computer Engineering and Science, Prince Mohammed Bin Fahad University, Khobar, Eastern Province, Saudi Arabia.

Graduated: May ,2018

Graduation project: Eye-writer, Implement the hardware and software to help the disabled children interact with technology by monitoring their eye movements. The hardware was implemented using Arduino, and the software was written in Java.

EXPERIENCE:

- Working as Network Security Analyst at Al Ittefaq Steel Products Company.

From: April 2023 - Now

- Worked as Business Support Analyst at Al Ittefaq Steel Products Company.

From: December 2020 to April 2023

- Worked as Network Engineer and IT support at Twareat Medical Care Company.

From: September 2018 to September 2019

- Internship at Saudi Aramco, ECC Network Management Group

From: July 2017 to August 2017

HONORS & AWARDS

- DEAN'S LIST LETTER For excellent academic performance during the spring semester 2014/2015
- Certificate of Achievement for Excellent Academic Progress from PMU.