Control of Multi-agent Reinforcement Learning Systems Under Adversarial Attacks

Neshat Elhami Fard

A Thesis

in the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

November 2022

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Neshat Elhami Fard**

Entitled: **Control of Multi-agent Reinforcement Learning Systems Under Adversarial Attacks**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________	Chair
Dr. Yuhong Yan	
______________________	External Examiner
Dr. Aditya Mahajan	
______________________	External to Program
Dr. Youmin Zhang	
______________________	Examiner
Dr. Amir Aghdam	
______________________	Examiner
Dr. Shahin Hashtrudi Zad	
______________________	Supervisor
Dr. Rastko R. Selmic	

Approved by ______________________

Dr. Jun Cai, Graduate Program Director

11/28/2022 ______________________

Dr. Mourad Debbabi, Dean

Gina Cody School of Engineering and Computer Science

Abstract

Control of Multi-agent Reinforcement Learning Systems Under Adversarial Attacks

Neshat Elhami Fard, Ph.D.

Concordia University, 2022

This Ph.D. dissertation studies the control of multi-agent reinforcement learning (MARL) and multi-agent deep reinforcement learning (MADRL) systems under adversarial attacks. Various attacks are investigated, and several defence algorithms (mitigation approaches) are proposed to assist the consensus control and proper data transmission.

We studied the consensus problem of a leaderless, homogeneous MARL system using actor-critic algorithms, with and without malicious agents. We considered various distance-based immediate reward functions to improve the system's performance. In addition to proposing four different immediate reward functions based on Euclidean, n-norm, and Chebyshev distances, we rigorously demonstrated which reward function performs better based on a cumulative reward for each agent and the entire team of agents. The claims have been proven theoretically, and the simulation confirmed theoretical findings.

We examined whether modifying the malicious agent's neural network (NN) structure, as well as providing a compatible combination of the mean squared error (MSE) loss function and the sigmoid activation function can mitigate the destructive effects of the malicious agent on the leaderless, homogeneous, MARL system performance. In addition to the theoretical support, the simulation confirmed the findings of the theory.

We studied the gradient-based adversarial attacks on cluster-based, heterogeneous MADRL systems with time-delayed data transmission using deep Q-network (DQN) algorithms. We introduced two novel observations, termed on-time and time-delay observations, considered when the data transmission channel is idle and the data is transmitted on-time or time-delayed. By considering the distance between the neighbouring agents, we presented a novel immediate reward function that appends a distance-based reward to the previously utilized reward to improve the MADRL system performance. We considered three types of gradient-based attacks to investigate the robustness of the proposed system data transmission. Two defence methods were proposed to reduce the effects of the discussed malicious attacks. The theoretical results are illustrated and verified with simulation examples.

We also investigated the data transmission robustness between agents of a cluster-based, heterogeneous MADRL system under a gradient-based adversarial attack. An algorithm using a DQN approach and a proportional feedback controller to defend against the fast gradient sign method (FGSM) attack and improve the DQN agent performance was proposed. Simulation results are included to verify the presented results.

Acknowledgments

My deepest pleasure is to extend my sincere gratitude to my supervisor, Prof. Rastko R. Selmic, for his invaluable advice, constant support, and patience while I pursued my Ph.D. program. I have always been inspired by his immense knowledge and great experience in my academic research and daily life. Without his unwavering support, I would not have been able to complete this work.

In appreciation of their time and feedback, I would like to thank my committee members, Dr. Aghdam, Dr. Hashtrudi Zad, Dr. Zhang, and Dr. Mahajan.

I also appreciate all the support I received from my family, especially my parents, who gave me the encouragement I needed throughout this process.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AUV	Autonomous Underwater Vehicle
BIM	Basic Iterative Method
CNN	Convolutional Neural Network
CPS	Cyber-physical System
CRN	Cognitive Radio Network
CS-DLMA	Carrier-sense Deep Reinforcement Learning Multiple Access
CSMA	Carrier-sense Multiple Access
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-network
DRL	Deep Reinforcement Learning
FGM	Fast Gradient Method
FGSM	Fast Gradient Sign Method
FTP	File Transfer Protocol
HRI	Human-robot Interaction
IFCS	Intelligent Flight Control System
IT	Information Technology
LSTM	Long-short-term Memory
MADDPG	Multi-agent Deep Deterministic Policy Gradient

MADRL	Multi-agent Deep Reinforcement Learning
MARL	Multi-agent Reinforcement Learning
MAS	Multi-agent System
MCS	Mobile Crowd-sensing
MDP	Markov Decision Process
ML	Machine Learning
MSE	Mean Squared Error
MTDDQN	Multi-source Transfer Double DQN
NN	Neural Network
PID	Proportional-integral-derivative
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
SARSA	State Action Reward State Action
SDN	Software-defined Networking
SU	Secondary Users
TL	Transfer Learning
TRPO	Trust Region Policy Optimization
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UMV	Unmanned Maritime Vehicle
URL	Uniform Resource Locator

List of Publications

[J1]. N. Elhami Fard and R. R. Selmic, “Adversarial Attacks on Heterogeneous Multi-agent Deep Reinforcement Learning System with Time-delayed Data Transmission"; in *Journal of Sensor and Actuator Networks (JSAN)*, vol. 11, no. 3, pp. 45, Aug. 2022 [2].

[J2]. N. Elhami Fard and R. R. Selmic, “Consensus of Multi-agent Reinforcement Learning Systems: The Effect of Immediate Rewards"; in *Journal of Robotics and Control (JRC)*, vol. 3, no. 2, pp. 115-127, Mar. 2022 [3].

[C1]. N. Elhami Fard and R. R. Selmic, “Data Transmission Resilience to Cyber-attacks on Heterogeneous Multi-agent Deep Reinforcement Learning Systems"; in Proc. *The 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Singapore, 2022 [4].

[C2]. N. Elhami Fard and R. R. Selmic, “Time-delayed Data Transmission in Heterogeneous Multi-agent Deep Reinforcement Learning System"; in Proc. *30th Mediterranean Conference on Control and Automation (MED)*, Vouliagmeni-Athens, Greece, 2022, pp. 636-642 [5].

[J3]. N. Elhami Fard, R. R. Selmic, and K. Khorasani, “A Review of Techniques and Policies on Cybersecurity Using Artificial Intelligence and Reinforcement Learning Algorithms"; submitted to *Frontiers in Artificial Intelligence*.

[J4]. N. Elhami Fard, R. R. Selmic, and K. Khorasani, “Public Policy Challenges and Technical Considerations for Autonomous Systems"; submitted to *IEEE Technology and Society Magazin*.

[C3]. N. Elhami Fard and R. R. Selmic, “Control of Multi-agent Reinforcement Learning Systems: The Effect of Neural Network Structure Manipulation"; to be submitted to *2023 IEEE International Conference on Systems, Man and Cybernetics (SMC), Maui, Hawaii, 2023*.

Chapter 1

Introduction

1.1 Literature Review

A system that reacts intelligently and flexibly to changing operating conditions and demanding processes is an autonomous system. These systems, which have been booming in civilian and military applications recently, can modify their behaviour in response to unpredictable events. The railway, shipbuilding, aircraft, and robotics industries have developed new self-driving or autonomous systems products.

In recent years, much attention has been paid to multi-agent systems (MAS) and their applications in various fields. The collective behaviour of social insects and animal groups in nature has inspired the design of MAS. The colonies of ants, school of fish, and flocks of birds are prominent natural examples where MAS mimic their behaviours [6]. Each MAS consists of several autonomous agents. Depending on the different tasks allocated to multiple agents, each agent decides to take appropriate action to solve the specific task according to different inputs [7]. Agents' communication and collaboration enable such a system to achieve complex goals that are impossible for individuals [8]. Cyber-attacks on MAS, including autonomous agents, are a significant issue that should be taken into consideration.

Cybersecurity refers to the process of protecting networks, systems, and applications against cyber-attacks. Attacks on digital information aim to gain access to, modify, or destroy sensitive data. Cybersecurity has become a significant challenge due to the attackers' creativity, and its importance is also seen in the military and defence industries.

1.1.1 Autonomous Systems

One of the main features of autonomous control systems is solving complicated optimization problems without human intervention in the presence of uncertainty in real-time [9]. Autonomous systems must have the recognition and discretion potency, evaluation and estimate authority, and decision-making power to independently perform various tasks in a dynamic environment [10]. These systems have a variety of sensors to understand environmental information so that they can distinguish, evaluate, and make decisions based on environmental data [10]. Therefore, the features of autonomous systems include inferring their own state, comprehending their surroundings, and performing self-navigation [11]. In addition to an autonomous single-agent system, the autonomous systems can be designed in the form of multi-agents to identify high-risk, hazardous, or inaccessible areas [12], [13].

Research and development of robotics and autonomous systems have led to significant advances in a wide range of areas, including unmanned ground vehicles (UGV), unmanned aerial vehicles (UAV), unmanned maritime vehicles (UMV), artificial intelligence (AI), and self-learning machines [14]. Navigation [15], [16], and 3-D path following [17] for autonomous underwater vehicles (AUV), AUV for oceanographic research [18], and rescue robots [19] have benefited from the development of autonomy. Autonomy in systems is directly related to the security of such systems. If an autonomous system is compromised, virtual and physical problems occur that lead to lost data, communication interruptions, and damage or loss of the system [20]. To ensure a proper operation, robot operating system security features are optimized utilizing encrypted communications and semantic rules [21]. Decision-making in all types of autonomous systems is a critical process and is a significant issue [22]. An always-present dilemma - "which decision should be made by a software system, and which one should be made by a human." Software developers for autonomous systems have a responsibility to answer the above question [23].

1.1.1.1 Autonomous Systems: Background

The first remote control system with limited autonomous capability was proposed in the 19th-century [24]. In the 1930s, primary types of the autonomous system were used for travel, including auto tillers to protect sailboats from wind and autopilot to preserve altitude and speed using a gyroscope [25]. In 1940, while developing automated range finders for anti-aircraft guns, Norbert Wiener introduced a new field, termed cybernetics [26]. In the 1950s, General Motors inspected and tested new cars driven by special auto constructions [25]. An autonomous rover system, which was able to walk through the corridors of a building and charge its battery by identifying a power outlet on the wall and plugin into it, was introduced in 1964 [27]. Digital control electronics were developed in the 1970s. AI technology created automated perception and cognition. With the integration of these technologies, significant advances have been made in autonomous systems to perform complicated operations without human intervention [27]. The use of the autonomous system has increased dramatically in various fields, so they have had an undeniable influence on society. Autonomous systems are widely used to decrease human labour while enhancing productivity and safety [28], [29].

- **Levels of Autonomy:** Human-robot interaction (HRI) defines the types of robot interactions, with each category requiring a different level of autonomy [30]. Developers should evaluate autonomy levels suitable for their robot according to the framework guidelines and investigate the effects of autonomy on HRI [30]. In general, and for all systems, autonomy has four levels. These levels are level 0: human operation, level 1: automation, level 2: semi-autonomous, and level 3: autonomous. At level 0, the system's control is always performed by one operator, e.g., manual driving requiring constant supervision of steer, brake, or accelerate systems. At level 1, an operator continues to control the system and is responsible for operating the system safely. However, at this level of autonomy, limited control over a specific function is left to the system itself, e.g., providing steering OR brake/acceleration support to the driver, such as lane centring, OR adaptive cruise control (requires supervising steer, brake, or accelerate systems constantly). Since the system at level 2 is semi-autonomous, it performs some of the defined tasks without the operator's intervention. The rest of the functions are carried out by the operator, e.g., providing steering AND brake/acceleration support to the driver, such as lane centring AND adaptive cruise control simultaneously (requires supervising steer, brake, or accelerate systems constantly). Finally, at level 3, the system performs all defined tasks without human interaction and is responsible for the safety and critical activities, e.g., driving the vehicle everywhere under all conditions (does not require a driver).

- **Automated vs. Autonomous Systems:** According to the Merriam-Webster dictionary, [31] autonomy is either "the quality or state of being self-governing" or "self-directing freedom and especially moral independence." Automatic is defined as "having a self-acting or self-regulating mechanism" [32]. The different definitions of automation are "the technique of making an apparatus, a process, or a system operate automatically," "the state of being operated automatically," and "automatically controlled operation of an apparatus, process, or system by mechanical or electronic devices that take the place of human labour" [33]. According to the definition in [34], autonomy is a capability in which unmanned systems can combine various features to achieve the systems' goals. These specifications are sensing and perceptions of the environment, analysis, communication, planning, decision-making, and ultimately acting and executing. It is worth mentioning that human operators set the objectives for the system.

  As illustrated in Figure 1.1, the authors of [35] have defined five steps to go through automation and achieve autonomy. These modifications commence with a simple and assisted

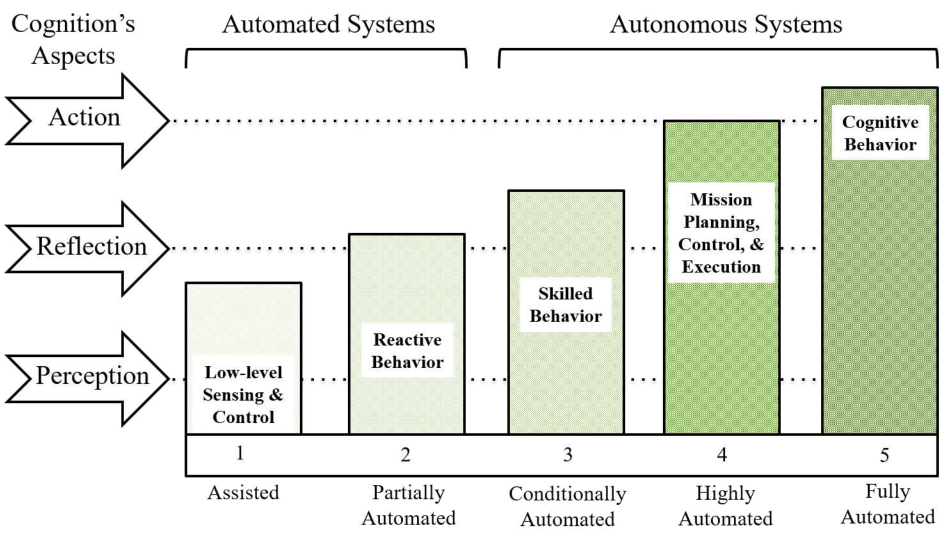


Figure 1.1: The steps of converting automated systems to autonomous ones.

behaviour, including "low-level sensing and control," and terminate with a full cognitive behaviour via a "very high degree of autonomy" [35]. An automated system lacks the ability to think about the consequences of its actions, and it is unable to modify a predefined sequence of activities. In contrast, an autonomous system has the ability to perceive and make decisions to accomplish its tasks based on purposes, skills, and learning experiences [35]. Despite the capacity to decide, these systems have significant shortcomings. Due to the existing drawbacks, autonomous systems in the aerospace industry are more applicable than road transportation. While autonomous flying is more reliable, it is predicted that automation driving will diminish fatal accidents by up to 90% [35]. Table 1.1 summarizes the technical specifications, advantages and disadvantages of automated and autonomous systems [36], [37], [38], [39].

1.1.1.2 Autonomous Systems: Significant Challenges

The design and implementation of any autonomous system face various challenges. Generally, the critical challenges posed by the advent of autonomous systems are technical challenges, professional responsibility difficulties, regulation hurdles, oversight challenges, public acceptance problems, and finally ethics challenges.

Table 1.1: Specifications, advantages and disadvantages of automated and autonomous systems, separately.

Specifications of Automated and Autonomous Systems	
	Features
Automated Systems (Automation)	• Sensing ability, • Execution ability to activate and perform operations manually regarding predefined rules, • Deterministic operation outcomes, • Need for manual intervention during operations.
Autonomous Systems (Autonomy)	• Sensing ability including advanced technologies, • Cognitive ability containing perceptual integration, pattern recognition, learning, reasoning, etc. to achieve self-defined goals, make strategy adjustment, allocate resources, etc., • Execution ability inclusive autonomous activation, and independent execution, • Adaptive ability to unpredictable environments, • Non-deterministic operation outcomes, • No need for manual intervention during operations; however, human should be the final decision maker per human factors design.

Pros and Cons of Automated and Autonomous Systems		
	Pros	**Cons**
Automated Systems (Automation)	• Reducing labour costs, • Accomplishing tasks faster, • Increasing the cheaper goods, • Optimizing the components' life, • Avoiding boring, and repetitive jobs, • Giving customers greater choice of goods, • Enhancing the use of available equipment, • Increasing safety, and decreasing risk of human error, • Preventing theft of the components or the entire system, • Improving productivity and increasing efficient production, • Supporting less experienced operators to perform efficiently, enhancing labour productivity, and increasing wages.	• Leading to unemployment, • Increasing monopoly power, • Leading to lack of empathy with events, • Decreasing quality of life due to deal with computers, and loss of human interaction.
Autonomous Systems (Autonomy)	• Saving time and spending it on other tasks, • Enhancing safety due to the lack of human error, • Increasing the use of these systems by people with disabilities, • Reducing the rate of collisions and crashes in mobile autonomous systems due to possess various sensors.	• Being extremely expensive, especially after the first release, • Leading to unemployment, which is catastrophic for the economy, • Worrying about computer crashing or malfunctioning, and resulting in a major collision.

- **Technical:** In addition to simulation and real-world trials, new validation and verification techniques are needed to ensure the safety and reliability of autonomous systems.
- **Professional Responsibility:** The right standards and regulations are required to create a culture alongside the generated challenges.
- **Regulation:** A fast, responsive, and advanced monitoring system is needed to communicate with regulators of various sectors to ensure that regulations for properly implementing standards are carefully considered and enforced.
- **Oversight:** A surveillance expert must consider the advantages of an autonomous system if there are uncertainty and risk in the mentioned system.
- **Public Acceptance:** Before deploying a new autonomous system, trust and public acceptance must be established between system users and service providers.
- **Ethics:** There should be mechanisms in the process design of autonomous systems that can have a collective, effective, and transparent decision-making to solve moral uncertainty, address the absence of human oversight, and inform about extending autonomous systems [40].

Additional challenges facing autonomous systems are described below.

- **Challenges Based on Autonomous Systems Capabilities:** In [41] the authors classify the types of autonomous systems based on their capabilities so that they can identify the varieties of related challenges. According to [41], autonomous systems capabilities are classified into: interaction with a real environment, HRI, motion planning and execution, and autonomous system perception. The resemblance of the challenges for the interaction with a real environment is the interaction with the unstructured environment and an approach to describe it so that autonomous systems can use it effectively. The challenge for HRI capability is that an effective standard interface for human-machine communication has not been defined in the military environment so far. Based on research in [41], the principal challenge of motion planning and execution capability is to describe the level of least intervention of human supervision in the cooperation attempt. Finally, another challenge related to autonomous system perception is to describe the environment that can be more similar to human perception. Consequently, the authors acknowledged that a solution to the mentioned challenges and better comprehension of autonomous systems and autonomous system-human interaction might be a common ontology for the autonomous system operationalization [41].

- **Challenges to the Establishment of Autonomous Systems:** Challenges to establishing autonomous systems include technological constraints, economic limitations, and also ethical and legal concerns [42], [43]. There are numerous debates regarding using lethal autonomous systems during the war. The entirety of these arguments includes accepting responsibility for the failure of an autonomous system, violation of war rules, non-acceptance of ethical robots to monitor human fighters, and the presence of a problem with assigning a specific goal for each independent system.

- **Resilience Challenges for Autonomous Systems:** In [44], two resilience challenges for autonomous systems are discussed. Resilience by design means that the developed cyber systems should be robust to large-scale quantifiable perturbations, and resilience by reaction, which is used in the cyber systems during a disturbance at run-time and strengthens the system by a *bounce back* ability.

- **Challenges of Military Ground-based Operations:** In [45], the authors have explored other challenges in military ground-based operations. They claim that the ground-based operations of the military, which depend on satellite communications, encounter significant challenges, namely, vulnerability against hacking, which leads to misconceptions and ultimately receiving false coordinates. Moreover, there are problems in real-time data processing and networked communications.

Table 1.2: Opportunities and risks of emerging autonomous systems in military, road logistics, and healthcare.

	Opportunities and Risks of Autonomous Systems for Royal Netherlands Army (RNLA)
Opportunities	• Generating more reliable and quicker situational consciousness and perception, • Extending the ability, persistence, and stability of operations, • Diminishing the physical and cognitive loads of soldiers, • Allowing and enabling the simultaneous execution of tasks for effective and useful actions, • Enhancing the speed of the OODA (Observe, Orient, Decide, Act) loop, • Defending and protecting the force.
Risks	• Communication signals applied by autonomous systems are vulnerable and assailable to cyber-attacks, namely hacking, blocking, and disrupting system performance, • How data is interpreted, and practical information is provided, is extremely tough for autonomous systems to simplify trust in decision-making, • How autonomous systems work, and how to solve their upcoming problems, is incomprehensible for defence communities, • Lack of development of various machines and moving towards autonomy due to insufficient trust in autonomous systems, • Operators' overconfidence on autonomous systems, • Creating possible new tensions between traditional soldiers and new technical experts and data scientists, • Changing of training requirements, education, careers, and the type of work which soldiers engage in, as well as, exchanging leadership positions due to generating autonomous systems, • The integration of autonomous systems and the possibility of replacing machines in some individuals or units will have an impact on the training of the armed forces, • In some autonomous systems, such as the killer robot, a negative public perception leads to a lack of consideration for the benefits of automation and autonomy and thus considering the human control, • Authoritarian and rebellious governments that care less about moral considerations may reinforce the autonomous systems and use them in dangerous inhumane ways, • Since those who use autonomous systems are able to distract and deny responsibility for attacks, assigning individuals to use autonomous systems is a challenging task, • Opponents and adversaries may have many moral and legal restrictions on the reproduction and use of autonomous systems in all areas, • The creation of autonomous systems may lead to an arms race so that powerful countries can show and use their potential.
	Opportunities and Risks of Autonomous Systems in Logistics
Opportunities	• Creating new job opportunities for the elderly, people with disabilities, and the underprivileged by integrating them with autonomous systems, especially for jobs like autonomous truck driver, • Reducing shipping operation costs up to 40% by eliminating drivers and using autonomous vehicles, • Save fuel by creating a network, including a leader and several followers. The leader determines the speed and direction of the rest of autonomous vehicles, thereby preventing additional fuel consumption.
Risks	• Tendency to make mistakes, errors, and possible situational misjudgments because of their computer nature, and consequently leaning to accident, • Increasing job loss, and consequently high unemployment rate.
	Opportunities and Risks of Autonomous Systems in Healthcare
Opportunities	• Opening up new commercial opportunities for the insurance and healthcare industries, • Reduction of emergencies leading to death for high-risk patients due to active monitoring.
Risks	• Rising privacy and security concerns due to weak security structure.

1.1.1.3 Opportunities and Risks of Emerging Technologies

The specific technologies, whose development or practical applications are possible to obtain or achieve, yet not obtained or achieved, are termed emerging technologies [46]. The five emerging technologies that are worth investing are: (*i*) sensing and mobility; (*ii*) advanced AI and analytic; (*iii*) post-classical compute and communications; (*iv*) augmented human; and (*v*) digital ecosystems [47]. Among the technologies mentioned above, AI as a pervasive process is used in the most critical technologies, namely adaptive machine learning (ML), edge AI, edge analytic, explainable AI, AI platform as a service (PaaS), transfer learning (TL), generative adversarial networks, graph analytic, and autonomous systems [47]. AI technologies provide an essential capability for autonomous systems. An autonomous system must be able to learn from past experiences and take action based on what it has learned [48]. The primary foundation of AI is ML, which creates computer knowledge through data, observations, and interacting with the world around them [49]. Therefore, the capability that AI gives to autonomous systems is provided through ML. In Table 1.2, opportunities and risks of emerging autonomous systems in the military, road logistics, and healthcare are discussed [50], [51], [52].

In addition to the opportunities presented in Table 1.2, practical (internal) and conditional (external) risks have raised. It should be noted that internal challenges include technical, personnel, and training issues, and also external risks contain perceptions, ethical, legal, military, and political subjects [50]. Challenges to autonomous systems' health include incorrect system decision-making due to faults from sensors, system misbehaviour due to faults from actuators, and system malfunction due to faults from electronic components. Autonomous systems encounter following various challenges to ensure their safe process. These systems must be designed to accurately sense their surroundings, deal with any anomalies in the environments, and make decisions consistent with the primary purpose of autonomous systems. Also, autonomous systems in teams, including humans and robots, may harm other group members by synchronizing with their actions and refusing disorders. Besides, an autonomous system must be able to detect any malfunctions and failures, early

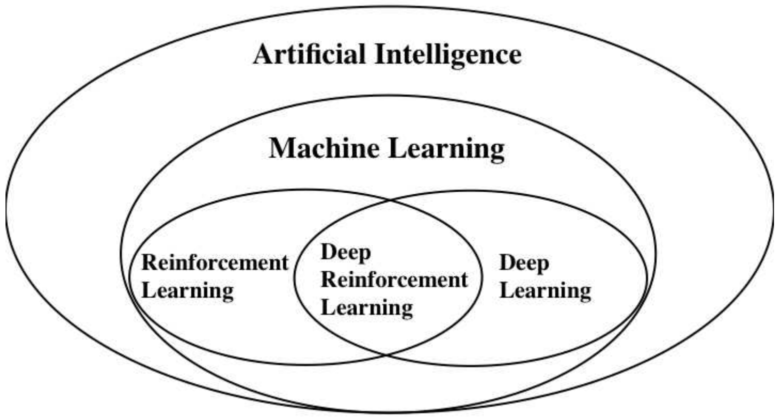


Figure 1.2: The defined relationships between AI, ML, RL, DL, and DRL [1].

warning, and respond quickly to identified system defects. The security challenges in autonomous systems are the security of hardware and software, as well as the security of communication links. Autonomous systems must be designed so that humans can interrupt them during emergencies. To achieve these conditions, task priorities, operations, interruption frequency, and timings must be considered; however, the main challenge of HRI is the complete identification of human kinds. This challenge is due to the diversity, complexity, uncertainty, and emotional expression of human beings [53].

1.1.2 Cybersecurity Using Artificial Intelligence and Reinforcement Learning Algorithms

The intelligence that has appeared in man-made machines and enables devices to imitate human behaviour is AI. AI is different from human and animal intelligence, which is intertwined with consciousness and emotions [54], [55]. As a subset of the AI method, ML has been used to improve the performance of devices, utilizing statistical techniques [56]. Reinforcement learning (RL) algorithms, which are used to solve numerous sequential decision-making problems, as well as deep learning (DL) methods, which utilize multi-layer neural networks (NN) for learning large volumes of data, are subsets of ML [1], [57]. Deep reinforcement learning (DRL), as a combination of DL and RL, has been established to overcome the high-dimensional problems of RL algorithms [58]. Relationships between AI, ML, RL, DL, and DRL are illustrated in Figure 1.2. Using RL and DRL, intelligent devices can learn from the actions they take at every time step, similarly to humans who learn from experiences [59].

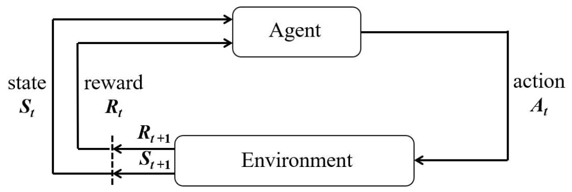
Cyber-physical systems (CPS) consist of cyber and physical subsystems, including various specifications, e.g., complexity, dynamic variability, and heterogeneity. Types of CPS include smart grids, smart transportation, and smart industrial manufacturing [60]. One of the RL and DRL applications is in cyber-attacks on CPS, as well as cyber-attack detection [57], [61]. Since the CPS models have infinite state-space, conventional methods such as cross-entropy are unsuitable for detecting their defects. RL, especially DRL, algorithms are superior to other methods in identifying CPS models' weaknesses [62]. An advantage of RL algorithms is that they can make accurate decisions automatically in an unknown environment by trial and error experiences [60]. According to [63], [64], [65], the cybersecurity tasks using DRL algorithms are intrusion detection, malware detection, privacy and security [59]. In [63], utilizing a labelled dataset, applications of various DRL algorithms have been offered for intrusion detection. In [64] a DRL architecture, including an adaptive cloud infrastructure, is proposed for intrusion detection. This DRL cloud intrusion detection system is robust to modern attacks and is able to maintain a balance between high accuracy and low false positive rate [64]. In [65] a DRL algorithm based on deep Q-networks (DQN) is presented to decrease the malware attacks in order to preserve the reliability, privacy, and security of the entire system.

There has been an increased effort to develop defence strategies and applications using RL and DRL algorithms. The authors in [66] have proposed three defence strategies using DRL. In detail, these methods increase the accuracy of the DRL agent that is attacked and, in contrast, take defensive positions against the dynamic actor-critic DRL jamming attacker. These strategies utilize a proportional-integral-derivative (PID) control, use an imitation attacker, and develop orthogonal policies. Another application of RL and DRL algorithms is in the MAS defence and attack. In MAS, cooperation between agents is a vital issue, where all agents, to accomplish their goals, learn efficient approaches. RL and DRL algorithms can be used to solve consensus problems. The authors of [67] have presented a multi-agent, deep deterministic policy gradient (MADDPG) algorithm for multi-agent defence and attack, containing rule-based attackers and DRL-based attackers accompanying DRL-based defence agents.

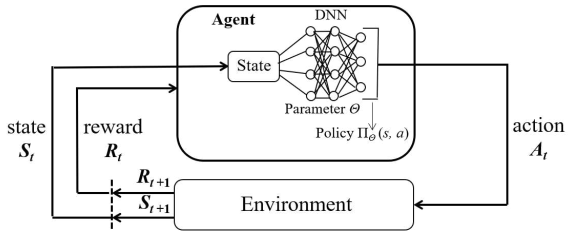
1.1.2.1 Cyber-physical System (CPS): Background

Cybernetics, during the development of automated range finders for anti-aircraft guns, was introduced by Norbert Wiener in 1940 [26], [68]. Wiener's proposed method is an adjustment and re-adjustment cycle by aircraft and an anti-aircraft gun. In this method, several observations are made to predict the future position of a flying aircraft during tracking, and the anti-aircraft gun's actions should be added to the previous forecast [68]. Both cyberspace and CPS terms are derived from cybernetics, respectively. Cyberspace was introduced by William Gibson in 1982 [69]. Based on Gibson's theory, cyberspace is a real non-space world identified by the ability to present virtually as well as interact with individuals through icons, waypoints, and AI [70]. Afterward, and for the first time, Helen Gill proposed the concept of CPS at the United States National Science Foundation (NSF) CPS workshop in 2006 [69], [71]. As a computer system, the mechanisms of a CPS are controlled and monitored using computer-based algorithms.

RL usage has flourished over the past decade. For the first time, Thorndike [72] introduced RL in 1898 with an experiment on cat's behaviour, and other researchers have further supplemented it over the years [73], [74]. In general, an agent of an RL algorithm receives a state from the environment and performs an action related to the received state at time step t. Then, according to the completed action, the agent gets a reward in the form of compensation or punishment from the environment at time $t+1$. The goal of the algorithm is to maximize the cumulative reward [75]. Figure 1.3a illustrates the RL structure [75]. DL, introduced by Walter Pitts and Warren McCulloch in 1943, is a computer model based on the human brain's NN and is a combination of algorithms



(a) A single-agent RL structure including the interaction between agent and environment.



(b) A single-agent DRL structure including a deep neural network (DNN).

Figure 1.3: Structures of RL and DRL algorithms.

and mathematics [76]. Finally, in 2015, DRL was developed by integrating DL architectures, and RL algorithms [58]. Different types of NN are utilized to convert RL to DRL algorithm [77]. The DRL structure is displayed in Figure 1.3b [78].

1.1.2.2 Challenges of RL and DRL Algorithms in CPS

Using RL and DRL Algorithms, CPS, both in cyber-attack and cyber-defence, is hampered by challenges similar to any other method. The following are the most significant challenges to consider:

- Although many RL and DRL algorithms present superior performance, hyperparameters and reward function design highly influence their performance. As a result, there is a great deal of uncertainty in training the controller [79].

- Many RL and DRL algorithms operate based on Markov decision processes (MDP), which are fully observable in terms of the environment. For CPS, in particular, the existence of an observable environment is false due to inherent uncertainties in states. This type of system is modelled by utilizing a partially observable Markov decision process (POMDP) [79].

Despite the discussed challenges, RL and DRL algorithms are now widely used as powerful learning techniques in various fields, especially in cyber-attacks and cyber-defence. These applications in CPS are exemplified by the following.

1.1.2.3 RL and DRL Algorithms for Cyber-attack

Various cyber-attacks related to the RL and DRL algorithms for different CPS are studied as follows:

- **Cyber-attacks on Smart Grids:** Since the smart grid is a complex CPS as an electrical network, including various operations and energy measures, multiple attacks and any other disruption in this network can damage the entire system. Attacks must be detected early for a smart grid to operate securely and reliably [80]. Due to the system's vulnerability to malicious attacks [81], we first examine these types of attacks in the smart grid. Malicious attacks are divided into two categories: physical-attacks and cyber-attacks [82]. In physical-attacks, the components of the power grids are physically attacked and damaged. These components include transformers, transmitters and transmission lines, generators, and others. This attack can be detected by the topology and data processing modules in the control center. Unlike physical-attacks, in cyber-attacks, power grid components are not physically attacked, but the measurable data transmitted in the supervisory control and data acquisition (SCADA) systems are modified and compromised. A well-thought-out and sophisticated cyber-attack can manipulate the topology and data processing modules and completely mislead or disrupt the performance of the control center [83].

  The authors of [81] have proposed a coordinated topology attack to overload a critical line by misleading the control center. In this method, both physical-attack and cyber-attack are integrated and performed coordinated topology attacks using DRL algorithms. The functions of the coordinated topology attack are: first, the physical-attack interrupts the transmission line; second, it hides the cut line signal in the cyber layer (masking a physical tripped line); and finally, it generates a fake cut-off signal for the following transmission line (creating a cyber tripped line). There are two types of DRL algorithms in this method: one to determine the attack strategy to overload a critical transmission line; another one to select minimal attack resources to inject the cyber-attack with limited attack resources. For the aforementioned coordinated topology attack, the DRL algorithms are based on deep Q-learning algorithms [81].

- **Cyber-attacks on Software-defined Networking (SDN):** “Software-defined networking (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled, or programmed, using software applications [84]." When the SDN’s cyber-defence system is defined using RL algorithms, the cyber-attacker tries to modify or destroy the training process in the RL algorithm. In this case, the cyber-attacker can perform this change and destruction in two ways: first, forging and changing part or all of the reward signal; second, manipulation and modification of some states received from the environment [85]. Therefore, when the defender agent receives the wrong state and reward from the environment, it will not perform the correct optimal action in response to the inaccurate data. As a result, the system can be compromised.

- **Cyber-attacks on CPS:** The major challenge that sometimes arises is the unknown cyber-attacks that threaten any CPS. One advantage of known cyber-attacks over unknown ones is that recognizing the type of attack and realizing how to model it can provide the cyber-defence more reliably and securely. Since the type of unknown cyber-attacks is undetectable, an online cyber-attack detection method must perform an optimal and effective defence strategy in an instant according to the existing conditions. In this regard, it is better to use learning-based methods (e.g., RL and DRL algorithms [86]) against these attacks to learn the existing conditions in real-time and defend online.

1.1.2.4 RL and DRL Algorithms for Cyber-attack Detection and Defence Applications

The different types of cyber-attack detection and defence applications that apply RL and DRL algorithms for diverse CPS are presented as follows:

- **Cyber-defence of Smart Grids:** Considering the problems that occur for smart grids due to issues related to cyber-attacks raised in Subsection 1.1.2.3, the authors of [80] have formulated an online cyber-attack detection problem as a POMDP. They have provided a robust online detection algorithm based on model-free RL algorithms for detecting cyber-attacks that targets a smart grid. The RL cyber-attack detection algorithm incorporates two stages: learning and online detection. In the learning stage, the defender is trained by the state action reward state action (SARSA) algorithm (a model-free RL algorithm) and learns a Q-table. In the online detection stage, the learned Q-table in the learning stage is utilized to select the action with the lowest expected future cost. Since the proposed algorithm by the authors of [80] is general, it is able to detect both known and unknown attacks.

- **Cyber-defence of SDN:** The authors of [85] have proposed an RL method to have a suitable autonomous cyber-defence against attackers who intend to infiltrate and propagate throughout the entire SDN and compromise critical network servers. This proposed method is supposed to train two diverse RL agents. One agent is based on double deep Q-networks (Double DQN) RL algorithm to isolate the agents and preserve them from attacks. Another agent is the asynchronous advantage actor-critic RL algorithm to keep nodes uncompromised and reachable from the critical server. Therefore, this method allows RL agents to perform sub-optimal actions despite attacks on RL algorithms [85].

- **Cyber-defence of CPS:** The authors of [86] have introduced and developed a particular type of optimal online cyber-defence to combat unknown cyber-attacks in a CPS. The steps of the suggested method are as follows. First, innovative cyber-state dynamics have been created to assess the effects of the current cyber-attack and the defence strategy efficiently and dynamically in real-time. Since cyber-attack is unknown, and there is insufficient information about the cyber-state dynamics, to provide an optimal cyber-defence, an actor-critic NN architecture has been proposed to learn the optimal online cyber-defence strategy effectively. Finally, a novel actor-critic DRL algorithm has been implemented to improve the proposed method [86].

- **Cyber-defence of Autonomous Vehicle Systems:** An adversarial DRL algorithm is presented by authors of [87] in order to maximize the robustness of autonomous vehicle dynamics control when it is subjected to CPS attacks. In addition, the authors investigated state estimation during monitoring of autonomous vehicle dynamics in the presence of CPS attacks. Based on game theory, the authors of [87] analyzed how the autonomous vehicle responds to CPS attacks and how the attacker acts. Using inaccurate data feeding into autonomous vehicle sensor readings, the attacker could potentially disrupt the optimal safe spacing between vehicles and cause autonomous vehicle collisions. As a defender, the autonomous vehicle attempts to minimize spacing errors so that it is resistant to attackers' destructive actions. Given the possibilities of manipulating data values and that the autonomous vehicle has no idea how the attacker is planning to attack, each player learns the expected spacing error using long-short-term memory (LSTM). An RL algorithm processes the error after being fed by the player. The attacker's RL algorithm attempts to make the attack as successful as possible, while the autonomous vehicle's RL algorithm explores the optimal action to minimize the spacing error. This is carried out with the Q-learning RL algorithm [87].

1.1.2.5 Other Applications of RL and DRL Algorithms in Cyber-security

- Generating a virtualized smart city network resource allocation that aims to assign virtual resources to a specific user, optimally, using a double duelling DQN RL algorithm [88].
- Using an actor-critic RL algorithm creates a mobile edge caching to maximize offloading traffic [89].
- Robustness-guided falsification of CPS can be done to find false inputs in these systems by integrating double DQN and actor-critic RL algorithms [90].
- Enhancing the robustness of the autonomous system against adversarial attacks to recognize corrupted measurements and decrease the effects of malicious errors using trust region policy optimization (TRPO) RL algorithm [91].

- Producing a secure offloading in mobile edge caching that aims to learn a policy for a mobile device to securely offload data to edge nodes against jamming and intelligent attacks using DQN RL algorithm with hot-booting TL methods [92].

- Creating an anti-jamming communication scheme for cognitive radio networks (CRN) in order to derive an optimal frequency hopping policy for CRN secondary users (SU) to overcome intelligent jammers based on a frequency-spatial anti-jamming game using DQN RL algorithm including convolutional neural network (CNN) [93].

- Developing other anti-jamming communication technologies to expand the previous item. In this scenario, jammers can use a variety of channel-slot-based architectures, including the recursive CNN algorithm and DQN RL algorithm [93], [94].

- A spoofing detection method is applied in wireless networks to determine the optimal authentication threshold using a Q-learning and Dyna-Q RL algorithm [95], [96].

- Hotbooting Q-learning and DQN RL algorithms are implemented as part of the development of mobile offloading for cloud-based malware detection, which will enhance detection accuracy and speed [97], [98].

- By developing a protected mobile crowd-sensing (MCS) system using the DQN RL algorithm, the payment policy is optimized to enhance the sensing performance facing faked sensing threats by forming a Stackelberg game [99].

- Using the DQN RL algorithm, an automated uniform resource locator (URL) based phishing detection is produced that identifies malicious websites (URLs) [100].

1.1.2.6 AI Algorithms for Defence Applications

There are diverse AI applications for the defence category, which are classified as follows [101], [102]:

- **Intelligence, Surveillance, and Reconnaissance:** AI has automated the workforce of data analysts who have to spend a lot of time analyzing various data to make essential decisions. In this field, multiple projects have been developed that are integrated with AI to generate analytical tools. These projects are developing algorithms for multilingual speech recognition and translation in noisy environments, fusing 2-D images to create 3-D models, geo-locating images without the associated metadata, and building tools to infer a building's function based on pattern-of-life analysis [101], [102].

- **Logistics:** AI is used to predict aircraft maintenance in the military instead of repairing damaged parts. Information is extracted from various aircraft parts, such as the engine, through real-time sensors and entered into a predictive algorithm. By focusing on the input data, this algorithm detects which parts of the aircraft should be inspected or which parts should be replaced. The AI scheduled maintenance program is another method to inspect different parts of an aircraft by schedule [101], [102].

- **Cyberspace Operations:** Since the standard cybersecurity tools seek to find pre-known matches to detect malicious codes, malicious users only need to change a small part of the code to disrupt the defence. In contrast, AI cybersecurity tools are trained to find irregularities in wider patterns of activity within a network and strengthen the protection. AI cybersecurity tools must be able to "detect, evaluate, and fix software vulnerabilities" so that military systems are able to perform tasks and defence simultaneously. This operation must be completed before an attack can take place [101].

- **Command and Control:** The information collected by different sensors is gathered in one place by AI and used centrally to control the systems [101].

- **Semi-autonomous and Autonomous Vehicles:** Integrating AI with autonomous and semi-autonomous systems aims to understand the surrounding environment, identify obstacles, navigate, and interact with other agents as a single agent in a MAS [101].

1.1.3 Position Consensus of Multi-agent Reinforcement Learning Systems: The Effect of Immediate Rewards

Applications of RL algorithms have increased over the years and led to tremendous advancements in various fields of science and robotics [103], [104]. The RL algorithms have been used to solve numerous sequential decision-making problems and have encountered significant hurdles when dealing with high-dimensional environments [57]. To partially overcome high-dimensional problems and perform tasks that require policy control, DRL algorithms were generated by combining DL, and RL algorithms [58], [105], [106]. In the combined algorithm, DL enables RL to address those challenges [107].

One of the DRL applications is in the MAS control [108], [109]. The use of MAS stems from nature, where multiple agents have higher efficiency and competitiveness when acting together in groups. Individual agents collaborate and interact with the environment to achieve the best results [57]. The consensus control is one of the fundamental problems in MAS, where agents support a common decision or aim in the best interest of the entire system. The agents participate in a group decision-making process, called consensus decision-making [110], [111], [112]. In consensus control, the goal is to reach a global agreement on a value or state for all agents [112], [113].

To reach consensus, we use the RL actor-critic method for N homogeneous agents. An actor, as the policy structure, decides to choose the best action based on its perception of the environment [114]. A critic, as the value function structure, indicates what is right in the long term and evaluates the selected actions by the actor [114]. In this stage, the internal structures of agents' actors and critics are multi-layer NN.

Consensus control in MAS has been studied in various situations, e.g. distributed optimal consensus algorithm [115], distributed linear-quadratic regulator (LQR) consensus control for heterogeneous MAS [116] or consensus control under delayed information [117], and has been extensively investigated for different systems such as linear and nonlinear systems [118], [119], [120],

[121], [122], [123], [124], [125]. In recent years, the consensus problem for multi-agent reinforcement learning (MARL) systems has also been researched, e.g. an optimal bipartite consensus control framework designed for MARL systems including model-free structure [126]. An integrated, resilient, model-free, off-policy, distributed state-feedback control protocol for leader-follower MARL system with adversarial inputs to reach consensus on the leader's state is proposed in [127], where only the leader can communicate with real information. The rest of the agents use a distributed observer to estimate the leader's state. The MARL system without malicious agents is implemented in [128], and the sum of the cumulative rewards is calculated and maximized. Inspired by [127], [128], we study a position consensus and we propose immediate reward functions that increase the cumulative reward with the presence of malicious agents in a leaderless MARL system in this Ph.D. dissertation. All agents of the on-policy system can communicate with the environment and receive the related states.

In order to use RL techniques in MAS, the authors of [129], [130], [131] have proposed actor-critic algorithms where the actor part is utilized for training of decentralized policies corresponding to each agent. Critic part is used for learning centralized value function including all agents information. In this Ph.D. dissertation, we use the decentralized actor-critic method [128] and the multi-agent actor-critic algorithm under adversarial attacks proposed in [132]. The authors in [132] have shown that the algorithm introduced in [128] for the consensus of MARL systems is not robust to adversarial attacks. In this Ph.D. dissertation, we present results in the presence of malicious agents by changing the immediate reward function of the RL algorithm. We propose four different distance-based immediate reward functions to select the best one (based on the results) and study their effects on the MARL system's cumulative reward. Our analysis of distance-based immediate reward functions shows that if a distance metric (e.g., Manhattan distance, Euclidean distance, $n-$norm distance, or Chebyshev distance) provides a smaller value between the current position and the desired position compared with the existing distance metrics, then the defined immediate reward function based on that distance metric will improve the MARL system performance.

Though different immediate rewards have already been introduced for various RL algorithms, studies of their effects on the MARL systems, with and without a malicious agent, are lacking. Defining the reward function according to the overall system's objective is preferred, as noted in [133], [134]. As the objective of the presented system is to reach a position consensus, considering the distance between the current position and desired position plays a significant role in achieving the consensus. Therefore, the main advantage of the proposed method is that the defined immediate reward functions are formulated using various distance metrics, e.g., Manhattan, Euclidean, $n-$norm, and Chebyshev distances, and their superiority over the Manhattan immediate reward function (as already used in [132]) have been proven. The superiority of the proposed immediate reward functions leads to higher average cumulative rewards for each agent. In addition, there is a greater average cumulative team reward for the entire MARL system. The MARL system performance is improved by obtaining a higher cumulative team reward, causing a higher percentage of correct actions are executed over time to achieve consensus, resulting in a position consensus with a lower error rate.

1.1.4 Control of Multi-agent Reinforcement Learning Systems: The Effect of Neural Network Structure

We studied the behaviour of undirected, leaderless, homogeneous MARL systems, including N agents, in reaching a position consensus using the RL actor-critic approach in the presence of malicious agents. In this chapter, the malicious agents are considered part of the MAS in which they transmit improper data to neighbouring agents' critic units or perform damaging actions [3]. Subsequently, we sought an approach to decrease the malicious agents' harmful effects on the MARL system performance.

The use of MAS has received much attention in the last two decades [135], [136]. MAS have been used in a wide range of applications in science and engineering, from industry, military, and medicine to smart grids, internet, and social networks [137], [138], [139], [140]. There have been several control techniques for MAS proposed over the past decade for network flocking, consensus, and formation control including adaptive control, slide mode control, tracking control, and back-stepping schemes [141], [142], [143]. In this chapter, we have used one of the RL algorithms called the decentralized actor-critic RL method to solve the MAS consensus problem under internal adverse attack [128]. Unlike the introduced decentralized actor-critic RL method in this chapter, the various actor-critic approaches have been presented for MAS, including the actor sector for each agent's decentralized policies training and the critic sector for all agents' centralized value function learning process [129], [130], [131]. The reason for using decentralized critic sectors in this chapter is to utilize the results of paper [144] which shows that in the introduced MARL system, the agents with centralized critic units converge to the optimal action with less stability, whiles the agents with decentralized critic units converge to the optimal action robustly.

The existing challenge is that MAS, like other systems, can be exposed to all kinds of adversarial attacks. In this regard, MAS attack detection and security of MAS have been a research topic in the past few years [145]. Adversarial attacks can be in the form of inner and outer threats [146]. In this chapter, the adversarial attack appears as an internal threat. In other words, one or multiple agents receive the correct data from the neighbours, but they transmit incorrect information to the rest of the neighbouring agents [3], [132]. Therefore, efforts should be made to nullify or mitigate the destructive effect of internal malicious agents.

As an extension of MAS, the MARL system has used RL algorithms in the structure of agents. In this chapter, the utilized RL approach in each agent's architecture is a policy-gradient-based actor-critic algorithm. Multi-layer NN is used in this chapter as the internal structures of agents' actors and critics. We have used the actor-critic framework in this chapter because these algorithms are useful in various real-life applications, including robotics, power control, and finance, due to their ability to search for optimal policies using low-variance gradient estimates [147].

1.1.5 Adversarial Attacks on Heterogeneous Multi-agent Deep Reinforcement Learning System with Time-delayed Data Transmission

The RL algorithm is the process of learning, mapping states to actions, and ultimately maximizing a reward signal through the interaction of an agent with a specific environment [114], [148]. DRL is characterized by a combination of RL and DL algorithms, two subdivisions of ML [107], [149], [150]. The DRL's advantage is that it addresses the high-dimensional problems that RL algorithms encounter [58], [105], [107]. Q-learning, as a type of RL algorithm, learns action values in a specific state [114]. Despite Q-learning's technological advances, it has one major flaw– similarly to dynamic programming, Q-learning can only update data within a two-dimensional array $\{\text{state}\times\text{action}\}$ [151]. The DQN algorithm is introduced, which merges Q-learning, and DNN [58], [152]. To cope with the two-dimensional array problem arising from the Q-learning algorithm, the DQN has been used in a wide range of applications [153], [154], [155]. There are two main reasons for using the DQN algorithm instead of other DRL approaches in this work: (i) the stability in performing complicated tasks. The discussed stability is the consequence of utilizing randomly sampled experience replay and a target network; (ii) the ability to predict the Q-value function.

DQN algorithm has been used in data transmission between multiple agents. The TL approach is combined with the DQN algorithm, and a multi-source transfer double DQN (MTDDQN) is introduced in [156]. The MTDDQN is based on actor learning and enables the collection, summarization, and transfer of action knowledge by the RL agent between multiple agents [156]. Compared to [156], this Ph.D. dissertation uses one DQN agent in a cluster-based, heterogeneous, MAS for on-time and time-delayed data transmission.

Data transmission in MAS has been investigated in various scenarios and for linear and nonlinear systems [157]. For instance, periodic event-triggered output regulation for linear MAS by considering a leader-follower topology is proposed in [158]. An adaptive event-triggered consensus control of linear MAS with directed leader-follower topology in the presence of a cyber-attack that affects the control input without modification in communication topology is developed in [159].

A dynamic event-triggered asynchronous control integrating fuzzy models with directed topology is presented in [160]. A new adaptive event-triggered leaderless consensus control of nonlinear MAS, including directed topology, can be found in [161]. Moreover, some researches address data exchanges between linear and nonlinear systems as a heterogeneous MAS, e.g., a leaderless and a leader-follower consensus of heterogeneous second-order MAS on time scales using an asynchronous impulsive approach, is presented in [162]. The previous studies have shown that there is little research on data transmission within homogeneous or heterogeneous multi-agent deep reinforcement learning (MADRL) systems, and the majority of the research has been focused on linear and nonlinear MAS. A heterogeneous MAS based on carrier-sense multiple access (CSMA) that utilizes DRL algorithm in data transmission, termed carrier-sense deep reinforcement learning multiple access (CS-DLMA), is introduced in [163]. The CS-DLMA uses $\alpha-$fairness objective to measure system performance. Inspired by [163] and using CS-DLMA, we study time-delayed data transmission between agents of a leaderless MADRL system in this Ph.D. dissertation. The same study is carried out for a leader-follower MADRL system. Note that CSMA is an access control protocol in which an agent in the network checks the state of data channel for data transmission.

Cyber-attacks can happen to any system, especially those that transmit data. Various adversarial attacks pose a threat to ML algorithms and DL systems [164], [165]. The ML algorithms are misled by adversarial attacks that manipulate input data to undermine algorithm performance, access the ML model, and modify model behaviour [166], [167]. Therefore, it is important to study the effects of various attacks on ML algorithms [168]. This Ph.D. dissertation uses three types of gradient-based adversarial attacks, termed fast gradient sign method (FGSM) [169], fast gradient method (FGM) [170], [171], and basic iterative method (BIM) [170], [172] to investigate their effects on the DQN algorithm, and consequently, MADRL system performance. Paper [163] is devoid of any information on cyber-attack on the system. The authors of [173] have examined the adverse effects of FGSM attack on DRL-based traffic signal control for a single-intersection and multiple intersection cases; however, its effects are not investigated on sending and receiving data. Hence, the FGSM attack plus FGM and BIM attacks, that try to fool the NN, are considered in this dissertation to check their impacts on the data transmission robustness. The authors of [174] have used the discussed attacks to target the observation set provided by the RL algorithm environment; as respects, we have applied the three types of adversarial attacks to target the produced environmental state of the DRL algorithm.

There are various defence techniques for ML algorithms, and these adversarial defence methods are used to improve the robustness of a designed model [175]. Among all the presented methods [176], [177], [178], [179], the best defence procedure occurs when the adversarial examples are fed to the NN training process [171]. In this regard, we use the worst perturbation as the input of the NN to train the model and reach the robustness against malicious attacks.

This Ph.D. dissertation scrutinizes adversarial attacks issues facing ML algorithms and studies the time-delayed data transmission robustness due to three types of gradient-based malicious attacks– FGSM, FGM, and BIM adversaries– between agents of a cluster-based, heterogeneous MADRL system. This study shows how the leaderless or leader-follower MAS perform due to time-delayed data transmission. After occurring attack for a leader-follower MADRL system, this Ph.D. dissertation presents two adversarial attack defence approaches against gradient-based attacks. This dissertation does not study the detectability of attacks, but how to reverse the attacks.

A potential application of this research is to use the proposed system in smart grids to make the grid more reliable, secure, and efficient. Moreover, by converting static agents to mobile agents and considering relevant contributions to the novel architecture, this system can be used for data transmission between the agents of all types of multi-agent autonomous vehicles, e.g., multi-rescue robots.

1.1.6 Data Transmission Robustness to Cyber-attacks on Heterogeneous Multi-agent Deep Reinforcement Learning Systems

We studied the behaviour of undirected, leaderless (or undirected, leader-follower) heterogeneous, MADRL system in reaching a cluster-based consensus, using a DQN algorithm in the absence and presence of an FGSM adversarial attack. We designed an integrated method of classical and learning-based control to investigate the MADRL system behaviour before and under adversarial attack in various scenarios. The designed proportional controller assists the DQN learning process in defending against adversarial attacks and reaching data transmission consensus for each cluster.

A lower loss and a higher DQN average reward after starting training are advantages in using a proportional controller along with the DQN algorithm. Achieving acceptable values in a shorter period of time reduces various costs (e.g., the cost of electricity consumption), saves time, and reduces equipment deterioration.

A PID controller can have unstable controlled process inputs if its gains (K_p, K_i, K_d) are set incorrectly. Therefore, proper selection of controller parameters is a significant issue [180], [181], [182], [183]. The authors of [184] showed that the proportional-integral (PI) controller can be automatically tuned by DRL methods. Moreover, in [185], [186], [187] the DQN algorithm is applied to design the controller parameters optimally. The authors of [185], and [186] designed a type-2 fuzzy PID controller, and a tilt fuzzy cascade controller, respectively. The parameters of the discussed controllers in [185], [186] are optimized using DQN algorithms under various operating conditions. Furthermore, the authors of [187] proposed a method of linear active disturbance rejection control and used the DQN algorithm to obtain controller parameters in real-time. Though DQN has been used with a variety of controllers in [185], [186], [187] and other research, this Ph.D. dissertation reverses the trend and uses a feedback control system (e.g., a proportional controller) to assist the DQN algorithm and defend against the FGSM adversarial attack for on-time and time-delay data transmission.

The authors' purpose of [188] is to compare the proposed feedforward feedback control system based on the DQN algorithm with a feedback control system using the same parameters. The introduced method in [188] improved the control system performance and solved the feedforward controller design for nonlinear systems' problems. However, in the current investigation, we compare the feedback control system based on the DQN algorithm with the pure DQN approach in a MADRL system under an FGSM cyber-attack and verify the proposed algorithm's effectiveness.

Various methods have been used to schedule and transfer data between agents of a MAS, e.g., different types of AI methods [189]. To transfer data between *N* heterogeneous agents of various *P* clusters, we used the DQN algorithm as a learning approach to control the data transmission. Consequently, we used a proportional controller on the DQN algorithm to support the DQN agent and cope with the FGSM malicious attack when data is transferred between the cluster-based MADRL system agents. The authors of [190] applied a proportional and derivative (PD) controller to the learning policy in order to stabilize the discussed RL policy and to compensate for the shortcomings of the gradient-based algorithm; however, this controller alone produces very high costs to control the quadrotor system in [190].

The proportional controller calculates the error between a measured process signal and desired signal value and performs the corrective action if necessary [191], [192]. In this Ph.D. dissertation, we use the estimated reward of the DQN algorithm as the desired output for the feedback control system. Moreover, we utilize the DQN environmental received reward as the measured output. Learning from the error signal in terms of estimated reward and environmental received reward causes the DQN agent to make a more reasonable decision and to choose the appropriate action.

1.2 Contributions

The contributions of this Ph.D. dissertation can be summarized in several categories as below:

- We introduced a cluster-based MAS structure including *N* agents and *P* clusters. We presented the average position consensus for cluster-based MAS, including clusters with a unique purpose such that all agents' goal of all groups is to reach a global agreement on the same position state. Afterward, we provided the average position consensus for cluster-based MAS containing clusters with various goals. In this regard, the agents' goal of each group is to reach a local agreement on a position state; however, the discussed agreement differs from the other clusters' settlement. We extended all calculations and simulation examples from 2-D space to 3-D space.

- We studied the behaviour of a leaderless, homogeneous MARL system in reaching consensus using a decentralized actor-critic method with and without malicious agents. We defined and proposed various immediate reward functions based on different distance metrics. These immediate reward functions can be used to calculate the cumulative reward for each agent and the MARL system. This work examines whether changing the immediate reward can improve the system's overall performance even with the destructive effects of a malicious agent. Suppose one of the distance metrics (e.g., Manhattan, Euclidean, $n-$norm, or Chebyshev distances) provides a smaller value between the current position and the desired position compared with the existing distance metrics. Consequently, the extracted immediate reward from the discussed distance metric generates a higher return cumulative reward for each agent and the MARL system as a whole. Hence, the criterion for measuring the MARL system's behaviour is based on various immediate reward functions.

  First, we studied the immediate reward function based on Manhattan distance proposed by [132]. Therefore, we proposed immediate reward functions based on Euclidean, n-norm, and Chebyshev distances. Additionally, we provided an algorithm to combine various immediate reward functions and used them based on the maximum returned value during each episode to enhance the agents' cumulative reward with and without malicious agents within the MARL system. Then, we proved the superiority of the Euclidean immediate reward function over the Manhattan immediate reward function. We showed the superiority of the Chebyshev immediate reward function over the Euclidean immediate reward function. Moreover, we showed that the combined immediate reward function outperforms other immediate reward functions.

- We provided a compatible combination of the mean squared error (MSE) loss function and sigmoid activation function in the malicious agent's NN structure to decrease the cumulative reward and increase the cumulative loss of the discussed adversarial agent in the MARL system. In such an event, the harmful effects of malicious agents on the MARL system performance are mitigated, and the MARL system efficiency in reaching the position consensus is improved. We proved the superiority of the MSE loss function in combination with the linear activation function over the integration of the MSE loss function and sigmoid activation function in terms of achieving higher cumulative reward and lower cumulative loss.

- We studied the time-delayed data transmission problem between agents in a cluster-based, heterogeneous, MADRL system under adversarial attacks. In addition to the leaderless MAS, we proposed a leader-follower MAS, such that the preassigned leader in each cluster communicates with the leader of other clusters as well as the agents of its own cluster. We considered two novel observations in data transmission, called on-time and time-delay observations, and we investigated their effects on the DQN loss and team reward. We proposed a novel immediate reward function that considers the package length, packet header size, and distance between neighbouring agents to improve the MAS performance in terms of approximated cumulative team discounted reward during time-delayed data transmission. We considered the FGSM, FGM, and BIM adversaries (gradient-based attacks) to attack the DQN algorithm. Then we investigated the effects of such attacks on MAS performance and time-delayed data transmission. We introduced two defence algorithms against the performed adversarial attacks. In the proposed defence methods, the DQN agent's DNN learns from a state that produces the maximum perturbation value and uses its negative feedback to improve the system performance during an adversarial attack.

- We designed a proportional controller for the DQN algorithm against the destructive effects of the FGSM adversarial attack. We considered on-time and time-delay data transmissions to investigate a defence algorithm, including the DQN method and feedback control system. Therefore, we proposed an algorithm wherein the DQN algorithm as the feedback control system process defends against the FGSM cyber-attack. We proved the superiority of using the proportional controller on the DQN learning process in achieving the higher average approximated cumulative team discounted reward for the MADRL system.

1.3 Summary and Dissertation Outline

This dissertation is organized into eight chapters as below:

- **Chapter 1:** This chapter presents the introduction, literature review, contributions, and outline of this dissertation. The corresponding parts of this chapter appear in [J1], [J2], [J3], [J4], [C1], [C2], and [C3].

- **Chapter 2:** This chapter represents a brief background on the research ahead, including RL, DRL, MDP, MAS, consensus decision making, adversarial attacks, and team organization.

- **Chapter 3:** This chapter briefly discusses the average position consensus of cluster-based, heterogeneous MAS. This chapter presents the average position consensus of clusters, including various goals as well as global goals in both 2-D and 3-D spaces. The simulation examples verify the calculations for all scenarios. This chapter aims to connect non-learning methods and learning approaches, especially RL algorithms, to reach a consensus.

- **Chapter 4:** This chapter studies the position consensus problem of a leaderless, homogeneous MARL system using actor-critic algorithms with and without malicious agents. The goal of each agent is to reach the position consensus with the maximum cumulative reward. We study the immediate reward function based on Manhattan distance. We propose three different immediate reward functions based on Euclidean, n-norm, and Chebyshev distances. We present a combination of various immediate reward functions that yields a higher cumulative reward for each agent and the team of agents. By increasing the agents' cumulative reward using the combined immediate reward function, we have demonstrated that the cumulative team reward in the presence of a malicious agent is comparable with the cumulative team reward in the absence of the malicious agent. The results of this chapter are published as [J2].

- **Chapter 5:** This chapter represents the control of a leaderless, homogeneous MARL system using actor-critic algorithms in the presence of a malicious agent as one agent of the MAS. The overall purpose of this chapter is to mitigate the adverse effects of the malicious agent to improve the MAS performance. This aim has been achieved by the malicious agent's NN structure modification. The corresponding parts of this chapter appear in [C3].

- **Chapter 6:** This chapter studies the gradient-based adversarial attacks on cluster-based, heterogeneous MADRL systems with time-delayed data transmission. The structure of the MADRL system consists of various clusters of agents. The DQN architecture presents the first cluster's agent structure. We consider three types of gradient-based attacks to investigate the robustness of the proposed system data transmission. Two defence methods are proposed to reduce the effects of the discussed malicious attacks. The theoretical results are illustrated and verified with simulation examples. The results of this chapter are published as [J1] and [C2].

- **Chapter 7:** This chapter investigates the data transmission robustness between agents of a cluster-based, heterogeneous MADRL system under the FGSM– a gradient-based adversarial attack. An algorithm is proposed using a DQN approach and a proportional feedback controller to defend against the FGSM attack and improve the DQN agent performance. The data transfer is carried out between agents of a MADRL system in a timely and time-delayed manner for both leaderless and leader-follower scenarios. The corresponding parts of this chapter appear in [C1].

- **Chapter 8:** In this chapter, a summary of the dissertation is given, and possible future works have been discussed.

1.4 Dissertation High-level Overview

In Chapter 3, an overview of MAS position consensus methods which are used in Chapters 4 and 5 is provided. Moreover, in Chapter 3, a preface of cluster-based MAS consensus approach for a set of heterogeneous agents which is utilized in Chapters 6 and 7 is presented, symbolically.

In Chapter 4, by considering an agent as an internal malicious agent in the MARL system, reaching position consensus has been examined with the help of Chapter 3 and an effort has been made to improve the performance of the MARL system in terms of higher cumulative reward.

In Chapter 5, the problem described in Chapter 4 is studied under the assumption that there is an internal malicious agent in the MARL system. In Chapter 5, a different method to improve the MARL system performance in terms of higher cumulative reward for position consensus, provided in Chapter 3, is proposed.

In Chapter 6, gradient-based adversarial attacks on cluster-based, heterogeneous MADRL systems with time-delayed data transmission are considered. Chapter 6 builds on the model presented in Chapter 3 and explicitly allows for communication between preassigned leaders across clusters.

In Chapter 7, gradient-based adversarial attacks are investigated for DQN algorithms and a linear feedback control system. The analysis generalizes the results derived in Chapter 6. Chapter 7 builds on the model presented in Chapter 3 and consequently in Chapter 6.

Chapter 2

Background

2.1 Introduction

This chapter deals with the background of our study. A brief history of RL algorithms, DRL approaches, and the MDP as a decision-making process is presented. Later, a concise description of MAS is explained. A brief explanation of consensus decision-making and team organization is provided.

RL has a large number of applications, from finance to robotics [193]. In recent decades, the use of RL and DL in solving various complex problems has increased significantly. Over the past few years, with DRL's creation, which is a combination of RL and DL, new solutions have been proposed to solve some of the complicated problems. In one kind of classification, there are two types of methods for solving RL and DRL decision-making problems. If a model of the environment and planning are used to address the issues as mentioned earlier, it is called a model-based method. Model-based is an approach to derive the rewards and optimal actions using a model of the transition dynamics [193]. In contrast, if a more straightforward trial-and-error approach is applied to obtain the rewards and optimal actions, without using the environmental model, it is termed a model-free technique.

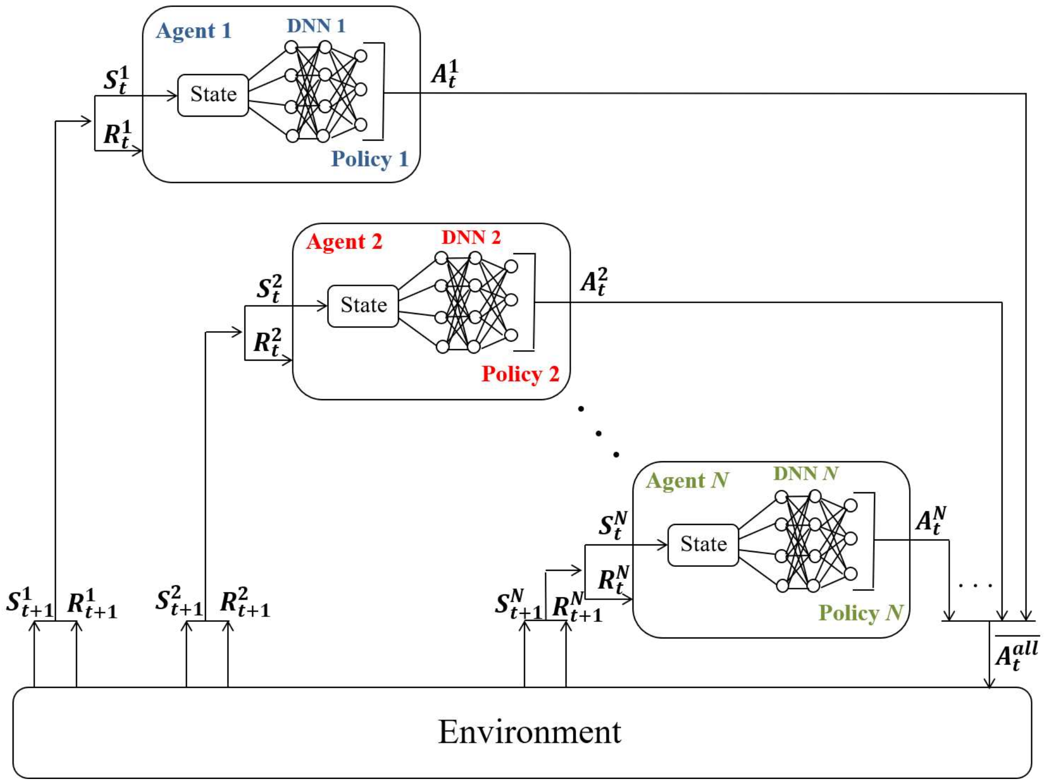


Figure 2.1: An illustration of the generic DRL structure of our heterogeneous MAS. At each time step t, each agent performs action A_t based on the learned policy. At time $t+1$, the environment returns a new state and reward to each agent.

As demonstrated in Figure 2.1, in the proposed multi-agent DRL, at each time step t, each agent receives state from the environment and takes action. Then, at every time step t, a joint action of all agents is obtained and returned to the environment. Afterward, each agent receives the next state and a delayed independent immediate reward. Finally, at every time step t, a joint immediate reward is calculated. According to Figure 2.1, using the current and next environmental states in combination with the existence of joint action, the state transition probability can be calculated to describe the environment's dynamics. It is possible to have an independent local policy for each agent; however, several agents can use the same policy at the same time, depending on the circumstances. The set of independent local policies for N agents is described as $Policy = \{\pi_1, \pi_2, ..., \pi_N\}$. The set of local policies for N agents when two or more policies are similar to each other is defined as $Policy = \{\pi_1, \pi_2, ..., \pi_\Upsilon\}$, where $\Upsilon < N$.

2.2 Reinforcement Learning (RL)

There are three branches of ML methods, namely, supervised learning, unsupervised learning, and RL. In supervised learning, there are input variables and an output variable, where an algorithm is used to learn the mapping function from the input to the output [194]. In this way, with new input data, the output variable for that new input data is predictable [194]. In contrast, in unsupervised learning, there are input data and no corresponding output variables. This learning method is applied to model the underlying structure or distribution in the data to learn more about the data [194]. RL is an intersection of various fields of science such as ML (computer science), optimal control (engineering), operations research (mathematics), bounded rationality (economics), classical/operant conditioning (psychology), and reward system (neuroscience) [75]. RL includes a complete, interactive, goal-seeking agent and an environment [75], which the agent learns from by trial-and-error interactions with it, and receiving rewards after choosing actions.

The choices made by the agent are actions. Besides, the basis for making the choices and the base for evaluating them are states and rewards, respectively [77]. The characteristics of a learning agent are the ability (i) to sense the environment's state, (ii) to take the relevant actions to affect that states, and (iii) to have the ultimate goals [75]. Thereby, any method that is well-designed to solve these three aspects of agent specifications is considered as RL [75]. One RL agent may contains four main parts as following:

- policy, which is a mapping from perceived states to actions, is the core of RL agent [75]. In other words, at every time step t, an agent perceives the state of the environment, and then the agent's policy decides to choose the best action based on its perception. Selecting the best action leads to find the optimal policy, $\pi^* : S \rightarrow A$, where π indicates the policy, S is a finite set of states of the environment, and A is a finite set of actions.

- reward signal has a crucial role in each RL system. When an agent transits from one state to the next, the environment returns a reward R. The agent's goal is to maximize the expected reward. Therefore, it can be said that the reward signal specifies the purpose of the RL system. The optimal policy π^* must be obtained from the policy π. In this regard, the optimal policy π^* can calculate the maximum expected reward.

- value function V^π, which is the most significant sub-component in RL agent, should be considered in decision-making. Value function shows that what is right in the long run [75]. It is worth mentioning that the state-action value function (Q function) is used to determine which action should be selected by the agent in the current state using a policy π [195]. The relationship between policy, reward, and value function is given in the following equations.

The total amount of reward including a discount factor $\gamma \in [0,1)$ is given by

$$R_t = \sum_{t=0}^{\infty} \gamma^t R(s_t), \tag{2.1}$$

where $R(s_t)$ is the reward for simply being in the state s at time t (immediate reward at time t).

Since the value function is the expected future reward, the purpose is to maximize the value function as the total reward over time. In this regard, finding the policy with the highest expected reward is required. To evaluate a given policy, the value function is utilized as

$$V^\pi(s) = R(s,\pi(s)) + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t,\pi(s_t))\right], \tag{2.2}$$

where $R(s_t,\pi(s_t))$ is the reward for being in the state s at time t, taking policy $\pi(s_t)$. The expectation operator in Equation (2.2) averages the stochastic transition model, resulting in

$$V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} p(s'|s,\pi(s)) V^\pi(s'). \tag{2.3}$$

The policy π can be extracted from the value function V as follows

$$\pi(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s') \right]. \tag{2.4}$$

The Bellman equation solution for each state s is used to obtain the optimal value function V^* as well as the optimal policy π^*:

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^*(s') \right]. \tag{2.5}$$

- model of the environment, that mimics the environment's behaviour and dynamics [75], is the fourth significant part of an RL system. The dynamics of the environment are transition and reward functions. If an agent uses only policy or value function, it is called a policy-based or a value-based agent, respectively. Moreover, an actor-critic is an agent including both policy and value function [75]. On the other hand, for solving RL problems, an agent which contains one or both of the policy and value function, is also divided into two categories; model-based and model-free [75] agent. The first approach uses models of the environment, whereas the second one applies trial-and-error to present a solution for RL problems [75]. Since model-based methods are based-on planning, they use simulated experience produced by a model. In contrast, model-free approaches rely on learning and apply real experience created by the environment [75]. However, the significant part for both model-free and model-based approaches is the estimation and calculation of value function [75].

The RL agent taxonomy is illustrated in Figure 2.2. In some classifications, the reward is one of the main parts of each RL agent. Reward signal is a numerical feedback that each agent receives as a result of the chosen action. This reward signal could be an actual reward or punishment. If it does the right task, then it gets a reward, and if it does a wrong job, it receives a punishment. These numerical rewards, which indicate what is right and what is wrong in environmental com-

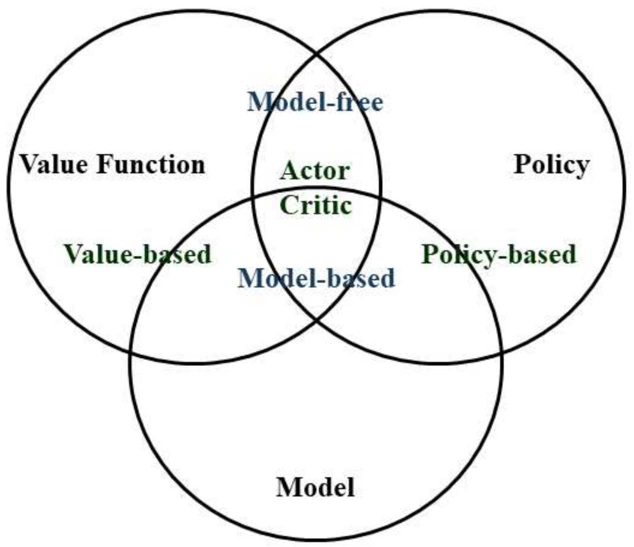


Figure 2.2: RL agent taxonomy.

prehension, often because of their delay, are called delayed rewards. The agent receives a delayed reward in the next time step to evaluate its former action.

One of the most critical challenges in RL is a trade-off between exploration, to take better future actions, and exploitation, to obtain rewards from past experiences [75]. Exploration finds more information about the environment, but exploitation exploits known information to maximize reward.

2.2.1 Reinforcement Learning-based Controller

Various types of controllers, such as PID controllers and intelligent flight control systems (IFCS), can be used to control a UAV [195]. To optimize the aircraft performance under normal conditions and to provide increased safety for the crew and passengers of an aircraft, the IFCS is designed [196]. The main benefit of IFCS is that it will allow a pilot to control an aircraft even under failure conditions (minor, major, hazardous, catastrophic) that would generally cause it to crash [196]. A PID is employed in stable environments because of its desirable and exceptional performance [195]. However, when the environment and dynamics are changeable, and various factors, such as wind and voltage sag, exist, the PID controller will not remain optimal. Therefore, an adaptable intelligent controller like an RL-based controller will be needed [195]. As an example, although the nonlinear controller based on feedback linearization has a better performance than the RL-based controller using a fitted value iteration algorithm for UAVs in [197], the design of the first one requires a lot of mathematical modelling knowledge. While for an RL-based controller, the quadrotor model can be approximated by input-output mapping via a non-parametric method. Remarkably, the fitted value iteration is an algorithm for approximating the value function of a continuous state (a finite sample of states) MDP. Note that the application mentioned in this subsection is based on UAVs and aircraft; however, it can be extended to other types of autonomous systems.

2.3 Deep Reinforcement Learning (DRL)

The combination of DL architectures and RL algorithms is a powerful DRL model that may be the solution to previously unsolved complex problems. In high-dimensional states, these complicated problems lead to finding a solution to a sequence of decisions [198]. The essence of RL is learning from the environment's feedback through trial and error to choose an action [198]. In other words, RL algorithms learn from successes and mistakes. However, in the process of trial and error, the agent may make a mistake in finding the proper action [198]. The characteristic of DL is the approximation of functions in high-dimensional problems using DNN in such a way that tabular methods are not able to provide exact solutions [198]. Since DL uses DNN to find approximations for large, complex, high-dimensional environments, the main difference between the RL and DRL algorithms is the use of DNN in the RL's agent structure. Hence, the DNN is used to convert RL to DRL algorithm [77]. By interacting with a complex, high-dimensional environment, DRL aims to learn optimal actions (learning from feedback) that maximize the reward for all states in every discussed environment [198]. The popularity of DRL is primarily due to its compatibility with current computers and its use in various applications [198]. Since DRL is looking for a solution for sequential decision-making problems, research in this field focuses on two basic types of applications. These applications include *robotic problems* and *games*.

2.3.1 Deep Reinforcement Learning-based Controller

When nonlinear function approximators are applied to estimate the value function, the RL approach will no longer be stable [58], [75]. Given this situation, the RL is combined with a specific type of function approximator, organized as a DNN, to form the DRL [58], [75]. This specified combination can directly map raw sensory outputs to the control signal [58], [75]. For example, UAVs' landing automatically on a ground marker is a significant problem. A DRL-based method is proposed to land these vehicles accurately on markers. At first, an attached camera to the UAV takes a low-resolution image. The ground marker position is identified from the received image, and finally, the UAV lands on it [199]. The recommended method comprises two DQN algorithms used as a high-level-control policy for landmark detection (marker detection) and vertical descent on a static pad, respectively [199]. Notably, the application mentioned in this subsection is based on UAVs; however, it can be expanded to other types of autonomous systems.

2.4 Markov Decision Process (MDP)

The MDP is a discrete-time stochastic control process that represents a fully observable environment for RL. It makes decisions to select the best action in a stochastic environment. Stochastic control, as a subfield of control theory, deals with the presence of uncertainty that drives the evolution of the system. It should be emphasized that in a fully observable environment, the agent can sense the entire state of the environment without any memory to make an optimal decision. An MDP is a 5-tuple $M = \langle S, A, p, R, \gamma \rangle$, where $S = \{s_1, s_2, \ldots, s_n\}$ is a finite set of states of the environment, $A = \{a_1, a_2, \ldots, a_m\}$ is a finite set of actions, $p(s'|s,a)$ is the state-transition probability matrix that agent starts in state s, takes action a, and ends in state s'. Moreover, $R : S \times A \rightarrow \mathbb{R}$ is a reward function, and $\gamma \in [0,1)$ is a discount factor. A factor that multiplies by the expected future rewards and shows the importance of future rewards versus immediate rewards is called the discount factor. The lower the discount factor is, the less significant future rewards are. Therefore, the agent tends to perform actions that result in immediate rewards. With $\gamma = 0$, the agent only cares about the immediate reward. Totaly, in RL, the MDP models the environment. In this process, the target is to calculate a policy π for mapping state to action that can maximize the expected sum of discounted reward [200] that is described as follows

$$J^{\pi} \equiv \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t, \pi(s_t))\right]. \tag{2.6}$$

2.4.1 Partially Observable Markov Decision Process (POMDP)

The POMDP is a combination of MDP and hidden states for modeling system dynamics [201]. In a partially observable environment, the agent can never see the entire state of the environment. In this situation, to make an optimal decision, one or more memory is needed. A POMDP is a 7-tuple $PM = \langle S, A, p, R, \Omega, O, \gamma \rangle$, where S, A, p, R, and γ are similar to their peers in MDP. Furthermore, $\Omega = \{o_1, o_2, \ldots, o_k\}$ is a finite set of observations, and O is a set of conditional observation probabilities where $O(o \mid s', a)$ is conditioned on the next state and the taken action. The POMDP is a mathematical model of uncertain decision-making. One should mention that in MDP and POMDP, state and action spaces are discrete or continuous. On the other hand, the signals to control and navigate UAVs are continuous. Thus often continuous state and action spaces are used for controlling these types of vehicles.

2.5 Multi-agent System (MAS)

If multiple intelligent agents interact with each other and ultimately pursue the same goal, they form a MAS. Sometimes it is challenging or impossible for a single-agent to solve multiple problems on their own, simultaneously. In these cases, the complex issues can be addressed by using a MAS and subdividing tasks between individual agents of the system [202]. In fact, in a MAS, the agents can perform multiple tasks in collaboration with each other due to their inherent ability to learn and make independent decisions. In any MAS, the metrics used for decision-making are sensed data from the environment (inputs), the knowledge of the system, and the determined goal for the mentioned system. Each agent in the MAS can communicate with any other agents in the network, and directly do an action on the environment based on the decision [7]. The main features of each MAS that are used in this Ph.D. dissertation are leadership, heterogeneity, delay consideration, topology, data transmission frequency, and mobility [7].

Leadership: In a MAS, an agent with a leadership role sets goals and tasks based on one global goal for the other agents. Leaderless or leader-follower MAS can be classified based on the presence or absence of such a leader. Hence, if in a MAS structure, each agent automatically decides to perform actions related to its purposes without the presence of a leader, it is a leaderless MAS. However, the leader-follower MAS allows the leader to determine actions for the rest of the agents [7].

Heterogeneity: A MAS can be classified into homogeneous or heterogeneous, depending on agents dynamics. The entire agents of a homogeneous MAS are identical and have similar dynamics, structures, and goals; whereas, heterogeneous MAS is a system including multiple agents in which agents' dynamics may be different or changeable [117].

Delay Consideration: In each MAS, each agent can be responsible for performing a specific task, and its information is essential to the entire system's survival. Therefore, there must be communication and data exchange between agents. During on-time (without time-delay) data transmission, the information of all agents is exchanged simultaneously; however, data conflict arises, and there is a possibility of data loss. In this regard, to prevent data loss, a delay can be considered to transfer data between agents of a MAS, which is called time-delayed data transmission. Given the importance of data, the system determines the priority of sending and receiving information between two agents. Therefore, the agent, including the most critical information, sends the data without time-delay, and the agent with the information of lower importance sends the data with time-delay [7].

Topology: A MAS topology depends on the location and relations between agents. The topology of a MAS has been chosen as dynamic or static topology. In dynamic topology, agents change position as they move; however, over the lifetime of an agent, a static topology maintains its position and relations [7].

Data Transmission Frequency: Agents in a MAS interact with the environment and transfer the received data in a time- or event-triggered manner. In time-triggered mode, each agent continuously interacts with the environment, gathers and transmits data to other agents at predefined intervals. In an event-triggered manner, the agent only interacts with the environment when a specific event happens. Later, the agent transmits the gathered data to other agents [7]. If control of a MAS structure is based on RL algorithms, then at each time step t, the agents receive the environment's state, and each agent must select a proper action in response. One step later, the agent gets a reward and a new state. Therefore, agents continuously sense the environment, collect data, and transfer the sensed data with other agents, in order of preference, in a time-triggered manner.

Mobility: If the agents of a MAS move through the environment and perform various tasks, such as identifying obstacles and detecting attacks, they fall into the category of mobile agents. The static agents are permanently positioned in the environment [7].

2.6 Consensus Decision-making

When agents in a MAS develop and agree to support a decision in the best interest of the entire system or common aim, they participate in a team decision-making process called consensus decision-making. Consensus control is known as one of the primary coordination problems in MAS [203]. The control algorithm's goal in consensus control is to reach a global agreement on a standard value or state for all agents [113].

2.6.1 Discrete-time Consensus Algorithms

Consider an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ including a set of vertices $\mathscr{V}$ and a set of edges $\mathscr{E} \subseteq \{(i,j) | i \in \mathscr{V}, j \in \mathscr{V}\}$ as a multi-agent system. At each time step t, the state updating in the discrete-time domain is performed by

$$x_i(k+1) = x_i(k) + \Upsilon \sum_{j \in \mathscr{N}(i)} a_{ij} \left(x_j(k) - x_i(k) \right), \tag{2.7}$$

where $x(k) \triangleq [x_1(k), \ldots, x_N(k)]^T \in \mathbb{R}^n$, and $\Upsilon = 1/deg(\mathscr{G})$ in which $deg(\mathscr{G})$ is the degree of graph $\mathscr{G}$. All the vertex degrees are figured out to find the degree of graph $\mathscr{G}$. The largest vertex degree is the degree of the graph $\mathscr{G}$. Furthermore, the adjacency matrix is described as $\underset{N \times N}{A} = [a_{ij}] \in \{0,1\}$, where

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathscr{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Moreover,

$$x(k+1) = P_\Upsilon x(k), \tag{2.9}$$

where P_Υ is the Perron matrix and is given by

$$P_\Upsilon = I - \Upsilon L. \tag{2.10}$$

In the Perron matrix of Equation (2.10), I and L are identity and Laplacian matrices, respectively.

2.7 Adversarial Attacks

A malicious attempt that perturbs a data point $x_0 \in \mathbb{R}^d$ to another point $x \in \mathbb{R}^d$ is an adversarial attack. In this case, the point x belongs to a particular target adversarial class. Suppose that there is an image with a feature vector x_0 of a dog image. An adversarial attack aims to create a different feature vector x that will be classified as another class specified by the attacker, e.g., a `cat`. This type of attack is *targeted* attack. The goal of each adversarial attack is not limited to pushing x_0 to a specific target class $\mathscr{C}_t$. Instead, it may pursue the goal of moving x_0 away from the original class $\mathscr{C}_i$. This type of attack is *untargeted* attack [204].

The move of a data point x_0 from $\mathscr{C}_i$ to $\mathscr{C}_t$ can be achieved in several ways. One of the common definitions in this field is to consider an operator $\mathscr{A} : \mathbb{R}^d \to \mathbb{R}^d$ such that $x = \mathscr{A}(x_0)$ is the perturbed data [204].

Definition 2.1 *(Adversarial Attack). Let $x_0 \in \mathbb{R}^d$ be a data point belong to class $\mathscr{C}_i$. Define a target class $\mathscr{C}_t$. An adversarial attack is a mapping $\mathscr{A} : \mathbb{R}^d \to \mathbb{R}^d$ such that the perturbed data*

$$x = \mathscr{A}(x_0)$$

is misclassified as $\mathscr{C}_t$ [204].

The additive model is a common adversarial attack model, in which linear operator $\mathscr{A}$ adds perturbation to the input [204].

Definition 2.2 *(Additive Adversarial Attack). Let $x_0 \in \mathbb{R}^d$ be a data point belong to class $\mathscr{C}_i$. Define a target class $\mathscr{C}_t$. An additive adversarial attack is an addition of a perturbation $r \in \mathbb{R}^d$ such that the perturbed data*

$$x = x_0 + r$$

is misclassified as $\mathscr{C}_t$ [204].

Depending on the type and complexity of the problem, one of the Definition 2.1 or Definition 2.2 can be extended for further analysis and introducing related defence algorithms.

2.8 Team Organization

In a MAS, when several agents form a group and define a specific goal for the created group, they have organized a team organization. In a team organization, each group's purpose is different from each agent's goal inside the group. There may be two or more groups in a team organization. To promote the decision-making process, each group, depending on its requirements, is able to request information from other groups' agents. Furthermore, the number of agents in a group is termed *team size*. The amount of data and information received from the environment is more considerable for larger team size; however, combining the data received by agents within the group requires more process [7].

Chapter 3

Average Position Consensus of Cluster-based Heterogeneous Multi-agent Systems

3.1 Introduction

This chapter provides a preface of the cluster-based MAS position consensus methods ahead. The difference is that no ML method is utilized in the current chapter to reach the agreement. This chapter presents the cluster-based average position consensus for two different scenarios in 2-D and 3-D (to show the results in higher dimensions and better clarification) spaces.

3.2 Methodology

In this chapter, agents of a MAS change position as they move. Hence, the proposed MAS topology has been chosen as dynamic topology since it depends on the location and relations between agents. As shown in Figure 3.1, the proposed MAS includes N agents in P various clusters. The sets of

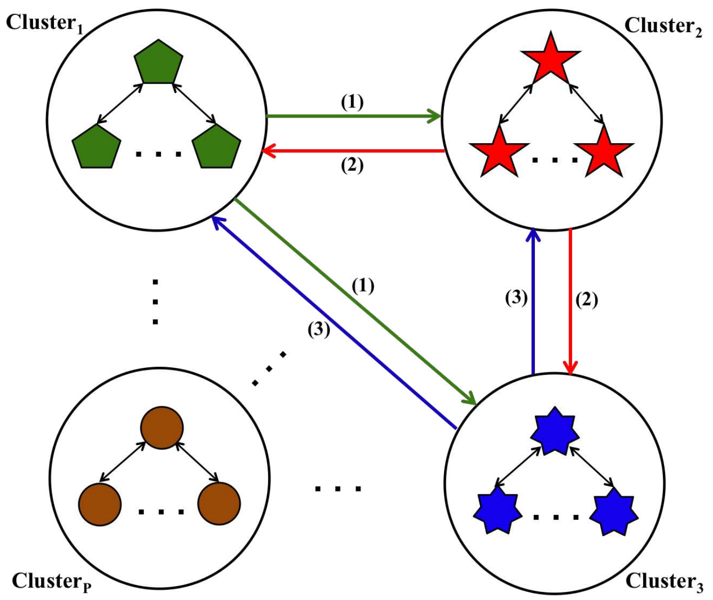


Figure 3.1: A generic illustration of the proposed cluster-based MAS structure, including N agents and P clusters, which other forms can be extracted from this structure.

agents and clusters are Agent = $\{Agent_1$, $Agent_2$, ..., $Agent_N\}$ and Cluster = $\{Cluster_1$, $Cluster_2$, ..., $Cluster_P\}$, respectively. The sets of various clusters in terms of agents are given below

$$\begin{cases} Cluster_1 & = \{Agent_1, Agent_2, ..., Agent_K\}, \\ Cluster_2 & = \{Agent_{K+1}, Agent_{K+2}, ..., Agent_L\}, \\ & \vdots \\ Cluster_P & = \{Agent_{M+1}, Agent_{M+2}, ..., Agent_N\}. \end{cases}$$

Agents of each cluster contain the same structure and same goal. Moreover, the agents' structure of one cluster is different from the agents' architecture of another group. Although each cluster has an independent goal, all P clusters could have a global goal. Hence, the clusters' goals could be different or the same.

In the graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ including a set of vertices $\mathscr{V}$ and a set of edges $\mathscr{E} \subseteq \{(i, j) | i \in \mathscr{V}, j \in \mathscr{V}\}$

(Figure 3.1), the adjacency matrix is described as $\underset{N\times N}{A} = [a_{ij}] \in \{0,1\}$, where

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathscr{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Moreover, the degree matrix is defined as $\underset{N\times N}{D} = diag(d_1, d_2, ..., d_N)$, where for $i \in \{1, 2, ..., N\}$ and the neighborhood $\mathscr{N}(i) \triangleq \{j \in \mathscr{V} : (i, j) \in \mathscr{E}\}$, the degree of i^{th} agent is given by

$$d_i \triangleq |\mathscr{N}(i)|. \tag{3.2}$$

Using the adjacency matrix $\underset{N\times N}{A}$ and degree matrix $\underset{N\times N}{D}$, the Laplacian matrix $\underset{N\times N}{L}$ is defined by

$$L \triangleq D - A, \tag{3.3}$$

where

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -a_{ij} & \text{if } i \neq j. \end{cases} \tag{3.4}$$

Regarding the above definitions, two types of position consensus based on different or global goals are explained in the following:

3.2.1 Condition for Position Consensus of Clusters with Various Goals

Since we considered a complete graph as a MAS including P clusters, the adjacency matrices for all P clusters, including N agents, are introduced in the set of Adjacency = $\{A_1, A_2, ..., A_P\}$ as follows:

$$\underset{K\times K}{A_1} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}, \quad \underset{(L-K)\times(L-K)}{A_2} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}, \ldots, \quad \underset{(N-M)\times(N-M)}{A_P} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}.$$

Furthermore, the set of Degree = $\{D_1, D_2, ..., D_P\}$ defines the degree matrices for all P clusters and N agents.

$$\underset{K\times K}{D_1} = \begin{pmatrix} K-1 & 0 & \cdots & 0 \\ 0 & K-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K-1 \end{pmatrix}, \quad \underset{(L-K)\times(L-K)}{D_2} = \begin{pmatrix} L-K-1 & 0 & \cdots & 0 \\ 0 & L-K-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L-K-1 \end{pmatrix}, \ldots,$$

$$\underset{(N-M)\times(N-M)}{D_P} = \begin{pmatrix} N-M-1 & 0 & \cdots & 0 \\ 0 & N-M-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N-M-1 \end{pmatrix}.$$

Note that the number of agents in one cluster is not necessarily equal to the number of agents in other clusters. According to the obtained adjacency and degree matrices for all P clusters, the graph Laplacian of P^{th} cluster is given by

$$\underset{(N-M)\times(N-M)}{L_P} = D_P - A_P = \begin{pmatrix} N-M-1 & -1 & \cdots & -1 \\ -1 & N-M-1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & N-M-1 \end{pmatrix}.$$

It should be emphasize that vector $v_P = \left[v_{P_1}, \ldots, v_{P_{N-M}}\right]^T \in \mathbb{R}^n$ is an eigenvector of Laplacian matrix L_P of eigenvalue λ_P if

$$L_P v_P = \lambda_P v_P. \tag{3.5}$$

Vector v_P satisfies the following relation for P^{th} cluster,

$$v_P^T L_P v_P = \frac{1}{2} \sum_{(i,j)\in\mathscr{E}} \left(v_{P_i} - v_{P_j}\right)^2 \geq 0. \tag{3.6}$$

Here, L_P is positive semi-definite. Since each cluster is a complete graph, then the graph Laplacian L_P has an eigenvalue equal to 0 ($\lambda_{P_1} = 0$), and the corresponding eigenvector $\mathbf{1} \in \mathbb{R}^n$. According to the connectivity of P^{th} cluster, the following is given for eigenvector $v_P = \mathbf{1}$.

$$v_P^T L_P v_P = 0. \tag{3.7}$$

Since the P^{th} cluster is a connected graph, then λ_{P_2} is strictly positive and $\lambda_{P_2} > 0$. Therefore, for the P^{th} cluster, the consensus can be achieved. The eigenvalues of Laplacian matrix L_P in ascending order are given by

$$0 = \lambda_{P_1} < \lambda_{P_2} \leq \cdots \leq \lambda_{P_{N-M}}.$$

Therefore, for complete graph $\mathscr{G}$ of P^{th} cluster, including $N-M$ agents, the Laplacian spectrum (the set of eigenvalues) is $\{0, N-M, \ldots, N-M\}$.

Note: If each group of agents is considered as an agent, then the adjacency, degree, and Laplacian matrices for P clusters are given by

$$\underset{P\times P}{A} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}, \quad \underset{P\times P}{D} = \begin{pmatrix} P-1 & 0 & \cdots & 0 \\ 0 & P-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P-1 \end{pmatrix}.$$

Hence, the Laplacian matrix is obtained as follows:

$$\underset{P\times P}{L} = D - A = \begin{pmatrix} P-1 & -1 & \cdots & -1 \\ -1 & P-1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & P-1 \end{pmatrix}.$$

3.2.1.1 Average Position Consensus of Clusters with Various Goals in 2-D Space

Assume that in 2-D space a random initial position state of i^{th} agent in P^{th} cluster is given by the pair of $(x_{P_i}(0), y_{P_i}(0))$. The i^{th} agent communicates with the corresponding neighbors to update its state at each time step t. Therefore, at each time step t, the state updating is performed by

$$\begin{cases} x_{P_i}(t+1) = x_{P_i}(t) + \Upsilon \sum\limits_{j\in\mathscr{N}(i)} a_{ij}(x_{P_j}(t) - x_{P_i}(t)), \\ y_{P_i}(t+1) = y_{P_i}(t) + \Upsilon \sum\limits_{j\in\mathscr{N}(i)} a_{ij}(y_{P_j}(t) - y_{P_i}(t)), \end{cases} \tag{3.8}$$

where $x_P(t) \triangleq \left[x_{P_1}(t), \ldots, x_{P_{N-M}}(t)\right]^T \in \mathbb{R}^n$, $y_P(t) \triangleq \left[y_{P_1}(t), \ldots, y_{P_{N-M}}(t)\right]^T \in \mathbb{R}^n$, and $\Upsilon = 1/deg(\mathscr{G})$ in which $deg(\mathscr{G})$ is the degree of graph $\mathscr{G}$ (multi-agent system). Moreover,

$$x_P(t+1) = P_\Upsilon x_P(t), \tag{3.9}$$

where P_Υ is the Perron matrix and is given by

$$P_\Upsilon = I - \Upsilon L. \tag{3.10}$$

In the Perron matrix of Equation (3.10), I and L are identity and Laplacian matrices, respectively. For P^{th} cluster the average vectors $x_{P_{avg}}$ and $y_{P_{avg}}$ are given by

$$\begin{cases} x_{P_{avg}} \triangleq \frac{\mathbf{1}^T x_P(0)}{N-M}\mathbf{1}, \\ y_{P_{avg}} \triangleq \frac{\mathbf{1}^T y_P(0)}{N-M}\mathbf{1}. \end{cases} \tag{3.11}$$

The error vectors are defined as follows:

$$\begin{cases} e_P^x(t) \triangleq x_P(t) - x_{P_{avg}}, \\ e_P^y(t) \triangleq y_P(t) - y_{P_{avg}}. \end{cases} \tag{3.12}$$

The purpose of the average consensus is to minimize the error during the time as below:

$$\begin{cases} e_P^x(t) \rightarrow 0 \ \ as \ \ t \rightarrow \infty, \\ e_P^y(t) \rightarrow 0 \ \ as \ \ t \rightarrow \infty. \end{cases} \tag{3.13}$$

3.2.1.2 Average Position Consensus of Clusters with Various Goals in 3-D Space

Suppose that in 3-D space a random initial position state of i^{th} agent in P^{th} cluster is given by the pair of $(x_{P_i}(0), y_{P_i}(0), z_{P_i}(0))$. The i^{th} agent communicates with the corresponding neighbors to update its state at each time step t. Hence, at each time step t, the state updating is performed by

$$\begin{cases} x_{P_i}(t+1) = x_{P_i}(t) + \Upsilon \sum\limits_{j \in \mathcal{N}(i)} a_{ij}(x_{P_j}(t) - x_{P_i}(t)), \\ y_{P_i}(t+1) = y_{P_i}(t) + \Upsilon \sum\limits_{j \in \mathcal{N}(i)} a_{ij}(y_{P_j}(t) - y_{P_i}(t)), \\ z_{P_i}(t+1) = z_{P_i}(t) + \Upsilon \sum\limits_{j \in \mathcal{N}(i)} a_{ij}(z_{P_j}(t) - z_{P_i}(t)), \end{cases} \tag{3.14}$$

where $x_P(t) \triangleq \left[x_{P_1}(t), \ldots, x_{P_{N-M}}(t)\right]^T \in \mathbb{R}^n$, $y_P(t) \triangleq \left[y_{P_1}(t), \ldots, y_{P_{N-M}}(t)\right]^T \in \mathbb{R}^n$, and $z_P(t) \triangleq \left[z_{P_1}(t), \ldots, z_{P_{N-M}}(t)\right]^T \in \mathbb{R}^n$. Furthermore, $\Upsilon = 1/deg(\mathscr{G})$ in which $deg(\mathscr{G})$ is the degree of graph $\mathscr{G}$. Moreover,

$$x_P(t+1) = P_\Upsilon x_P(t), \tag{3.15}$$

where P_Υ is the Perron matrix. For P^{th} cluster the average vectors $x_{P_{avg}}$, $y_{P_{avg}}$, and $z_{P_{avg}}$ are given by

$$\begin{cases} x_{P_{avg}} \triangleq \frac{\mathbf{1}^T x_P(0)}{N-M}\mathbf{1}, \\ y_{P_{avg}} \triangleq \frac{\mathbf{1}^T y_P(0)}{N-M}\mathbf{1}, \\ z_{P_{avg}} \triangleq \frac{\mathbf{1}^T z_P(0)}{N-M}\mathbf{1}. \end{cases} \tag{3.16}$$

The error vectors are defined as follows:

$$\begin{cases} e_P^x(t) \triangleq x_P(t) - x_{P_{avg}}, \\ e_P^y(t) \triangleq y_P(t) - y_{P_{avg}}, \\ e_P^z(t) \triangleq z_P(t) - z_{P_{avg}}. \end{cases} \tag{3.17}$$

The purpose of the average consensus is to minimize the error during the time as below:

$$\begin{cases} e_P^x(t) \to 0 \;\; as \;\; t \to \infty, \\ e_P^y(t) \to 0 \;\; as \;\; t \to \infty, \\ e_P^z(t) \to 0 \;\; as \;\; t \to \infty. \end{cases} \tag{3.18}$$

3.2.2 Condition for Position Consensus of Clusters with a Global Goal

Assume that all N agents in a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ belong to various P clusters (the agents' dynamics of each cluster are different from other clusters). It is considered that achieving a global goal requires all network agents are connected. Effectively, this is the same as the original model with one cluster including heterogeneous agents. This hypothesis aims to reach a global goal even though the agents' dynamics are different. The study of agents' dynamics is beyond the scope of this section, and the purpose is only to introduce the model for use in the following chapters. In this regard, the adjacency and degree matrices of Figure 3.1 are defined as below:

$$\underset{N\times N}{A} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}, \quad \underset{N\times N}{D} = \begin{pmatrix} N-1 & 0 & \cdots & 0 \\ 0 & N-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N-1 \end{pmatrix}.$$

Hence, the Laplacian matrix is obtained as follows:

$$\underset{N\times N}{L} = D - A = \begin{pmatrix} N-1 & -1 & \cdots & -1 \\ -1 & N-1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & N-1 \end{pmatrix}.$$

An eigenvector of Laplacian matrix L of eigenvalue λ is defined by vector $v = [v_1, \dots, v_N]^T \in \mathbb{R}^n$. Vector v must satisfy the following equation,

$$Lv = \lambda v. \tag{3.19}$$

Assuming for graph $\mathscr{G}$, vector v satisfies the following inequality,

$$v^T L v = \frac{1}{2} \sum_{(i,j)\in\mathscr{E}} \left(v_i - v_j\right)^2 \geq 0. \tag{3.20}$$

The Laplacian matrix L is positive semi-definite. Since $\mathscr{G}$ is a fully connected graph, the following equation is given for eigenvector $v = \mathbf{1}$.

$$v^T L v = 0. \tag{3.21}$$

Therefore, $\lambda_1 = 0$, and $\lambda_2 > 0$ is strictly positive. According to the values of λ_1 and λ_2, the consensus can be achieved for the agents of graph $\mathscr{G}$. The eigenvalues of Laplacian matrix L in ascending order are given by

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N,$$

and the Laplacian spectrum is $\{0, N, ..., N\}$ for complete graph $\mathscr{G}$ including N agents.

3.2.2.1 Average Position Consensus of Clusters with a Global Goal in 2-D Space

Assume that in 2-D space a random initial position state of i^{th} agent in graph $\mathscr{G}$ is given by the pair of $(x_i(0), y_i(0))$. The i^{th} agent communicates with the corresponding neighbors to update its state at each time step t. Therefore, at each time step t, the state updating is performed by

$$\begin{cases} x_i(t+1) = x_i(t) + \Upsilon \sum\limits_{j \in \mathscr{N}(i)} a_{ij}(x_j(t) - x_i(t)), \\ y_i(t+1) = y_i(t) + \Upsilon \sum\limits_{j \in \mathscr{N}(i)} a_{ij}(y_j(t) - y_i(t)), \end{cases} \tag{3.22}$$

where $x(t) \triangleq [x_1(t), \ldots, x_N(t)]^T \in \mathbb{R}^n$, and $y(t) \triangleq [y_1(t), \ldots, y_N(t)]^T \in \mathbb{R}^n$. Furthermore, $\Upsilon = 1/deg(\mathscr{G})$ in which $deg(\mathscr{G})$ is the degree of graph $\mathscr{G}$. Moreover,

$$x(t+1) = P_\Upsilon x(t), \tag{3.23}$$

where P_Υ is the Perron matrix. For graph $\mathscr{G}$ the average vectors x_{avg} and y_{avg} are given by

$$\begin{cases} x_{avg} \triangleq \frac{\mathbf{1}^T x(0)}{N}\mathbf{1}, \\ y_{avg} \triangleq \frac{\mathbf{1}^T y(0)}{N}\mathbf{1}. \end{cases} \tag{3.24}$$

The error vectors are defined as follows:

$$\begin{cases} e^x(t) \triangleq x(t) - x_{avg}, \\ e^y(t) \triangleq y(t) - y_{avg}. \end{cases} \tag{3.25}$$

The purpose of the average consensus is to minimize the error during the time as below:

$$\begin{cases} e^x(t) \to 0 \;\; as \;\; t \to \infty, \\ e^y(t) \to 0 \;\; as \;\; t \to \infty. \end{cases} \tag{3.26}$$

3.2.2.2 Average Position Consensus of Clusters with a Global Goal in 3-D Space

Suppose that in 3-D space a random initial position state of i^{th} agent in graph $\mathscr{G}$ is given by the pair of $(x_i(0), y_i(0), z_i(0))$. The i^{th} agent communicates with the corresponding neighbors to update its state at each time step t. Therefore, at each time step t, the state updating is performed by

$$\begin{cases} x_i(t+1) = x_i(t) + \Upsilon \sum_{j \in \mathscr{N}(i)} a_{ij}(x_j(t) - x_i(t)), \\ y_i(t+1) = y_i(t) + \Upsilon \sum_{j \in \mathscr{N}(i)} a_{ij}(y_j(t) - y_i(t)), \\ z_i(t+1) = z_i(t) + \Upsilon \sum_{j \in \mathscr{N}(i)} a_{ij}(z_j(t) - z_i(t)). \end{cases} \tag{3.27}$$

In the Equation (3.27), $x(t) \triangleq [x_1(t), \ldots, x_N(t)]^T \in \mathbb{R}^n$, $y(t) \triangleq [y_1(t), \ldots, y_N(t)]^T \in \mathbb{R}^n$, and $z(t) \triangleq [z_1(t), \ldots, z_N(t)]^T \in \mathbb{R}^n$. Furthermore, $\Upsilon = 1/deg(\mathscr{G})$ in which $deg(\mathscr{G})$ is the degree of graph $\mathscr{G}$.

Moreover,

$$x(t+1) = P_\Upsilon x(t), \tag{3.28}$$

where P_Υ is the Perron matrix. For graph $\mathscr{G}$ the average vectors x_{avg}, y_{avg}, and z_{avg} are given by

$$\begin{cases} x_{avg} \triangleq \frac{\mathbf{1}^T x(0)}{N}\mathbf{1}, \\ y_{avg} \triangleq \frac{\mathbf{1}^T y(0)}{N}\mathbf{1}, \\ z_{avg} \triangleq \frac{\mathbf{1}^T z(0)}{N}\mathbf{1}. \end{cases} \tag{3.29}$$

The error vectors are defined as follows:

$$\begin{cases} e^x(t) \triangleq x(t) - x_{avg}, \\ e^y(t) \triangleq y(t) - y_{avg}, \\ e^z(t) \triangleq z(t) - z_{avg}. \end{cases} \tag{3.30}$$

The purpose of the average consensus is to minimize the error during the time as below:

$$\begin{cases} e^x(t) \to 0 \;\; as \;\; t \to \infty, \\ e^y(t) \to 0 \;\; as \;\; t \to \infty, \\ e^z(t) \to 0 \;\; as \;\; t \to \infty. \end{cases} \tag{3.31}$$

3.3 Results and Discussion

This section provides the results of position consensus for 12 agents of four different clusters in a MAS. The number of agents in each cluster are

$$\begin{cases} 3\ agents \in Cluster_1, \\ 2\ agents \in Cluster_2, \\ 4\ agents \in Cluster_3, \\ 3\ agents \in Cluster_4. \end{cases}$$

3.3.1 Reaching Position Consensus of Clusters with Various Goals in 2-D Space

Each cluster of this MAS is considered as a sub-MAS and a complete graph. In this regard, the graph Laplacian of four various clusters of the mentioned MAS with an undirected topology are as below:

$$\underset{3\times 3}{L_1} = \underset{3\times 3}{D_1} - \underset{3\times 3}{A_1} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix},$$

$$\underset{2\times 2}{L_2} = \underset{2\times 2}{D_2} - \underset{2\times 2}{A_2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

$$\underset{4\times 4}{L_3} = \underset{4\times 4}{D_3} - \underset{4\times 4}{A_3} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix},$$

$$\underset{3\times 3}{L_4} = \underset{3\times 3}{D_4} - \underset{3\times 3}{A_4} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}.$$

Table 3.1: Initial and target positions of all 12 agents for four different clusters in 2-D space (various goals).

Agent / Cluster	1st Agent	2nd Agent	3rd Agent	4th Agent	Target Position
Cluster 1	$(1.1436, -1.5512)$	$(-0.9147, 0.9298)$	$(0.1798, 0.9019)$	—	$(0.2878, -0.0419)$
Cluster 2	$(-0.9833, 0.1383)$	$(0.3848, -0.3784)$	—	—	$(-0.2993, -0.1201)$
Cluster 3	$(0.3257, 0.1431)$	$(1.2963, 1.6050)$	$(1.0992, 1.3491)$	$(0.6532, -0.4484)$	$(0.8436, 0.6622)$
Cluster 4	$(-0.5051, 0.1684)$	$(-0.4760, -1.1205)$	$(-2.0516, 0.4007)$	—	$(-1.0109, -0.1838)$

Table 3.2: Initial and target positions of all 12 agents for four different clusters in 3-D space (various goals).

Agent / Cluster	1st Agent	2nd Agent	3rd Agent	4th Agent	Target Position
Cluster 1	$(1.5270, -0.1461, 1.2347)$	$(0.4669, -0.5320, -0.2296)$	$(-0.2097, 1.6821, -1.5062)$	—	$(0.6526, 0.2104, 0.2756)$
Cluster 2	$(0.6252, -0.8757, -0.4446)$	$(0.1832, -0.4838, -0.1559)$	—	—	$(0.4042, -0.6798, -0.3003)$
Cluster 3	$(-1.0298, -0.7120, 0.2761)$	$(0.9492, -1.1742, -0.2612)$	$(0.3071, -0.1922, 0.4434)$	$(0.1352, -0.2741, 0.3919)$	$(0.0904, -0.5881, 0.2126)$
Cluster 4	$(0.5152, 1.5301, -1.2507)$	$(0.2614, -0.2490, -0.9480)$	$(-0.9415, -1.0642, -0.7411)$	—	$(-0.0549, 0.0723, -0.9799)$

Using (3.19), the set of eigenvalues of each Laplacian matrix (Laplacian spectrum) are obtained as follows:

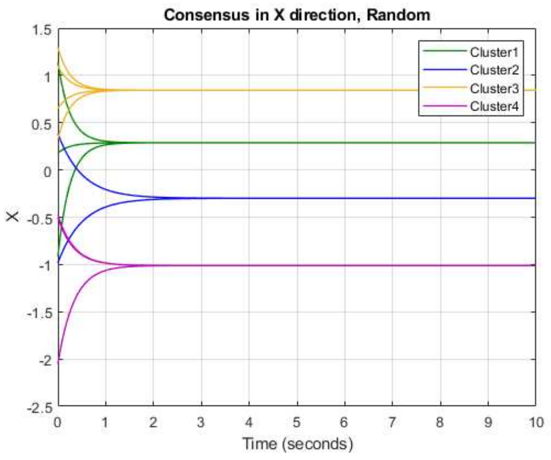
$$S_1^{\lambda} = \{0, 3, 3\},$$

$$S_2^{\lambda} = \{0, 2\},$$

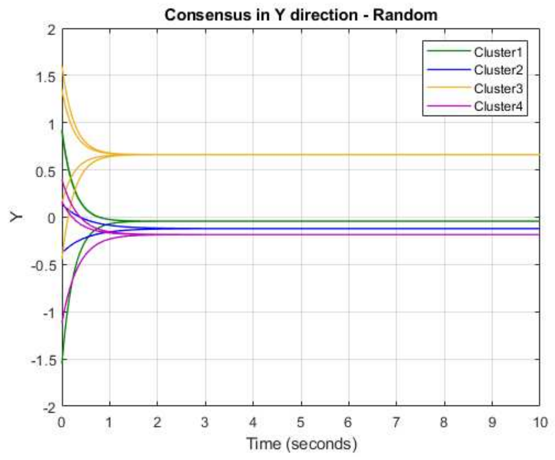
$$S_3^{\lambda} = \{0, 4, 4, 4\},$$

$$S_4^{\lambda} = \{0, 3, 3\}.$$

From the aforementioned eigenvalue sets, it is realized that each cluster is a connected graph. Therefore, reaching a consensus is possible for the agents of each cluster. Table 3.1 shows that the agents of each cluster converge to a specific target position. In this regard, all three agents of $Cluster_1$ converge to the point of $(0.2878, -0.0419)$. Moreover, the agents of $Cluster_2$, $Cluster_3$, and $Cluster_4$ converge to $(-0.2993, -0.1201)$, $(0.8436, 0.6622)$, and $(-1.0109, -0.1838)$, respectively. Figures 3.2-3.4 illustrate reaching consensus of 12 agents in four different clusters in 2-D space.



(a) Cluster-based consensus on $x-$direction.



(b) Cluster-based consensus on $y-$direction.

Figure 3.2: 12 agents of four different clusters reach consensus on $x-$direction and $y-$direction with random initial x and y during 10 seconds.

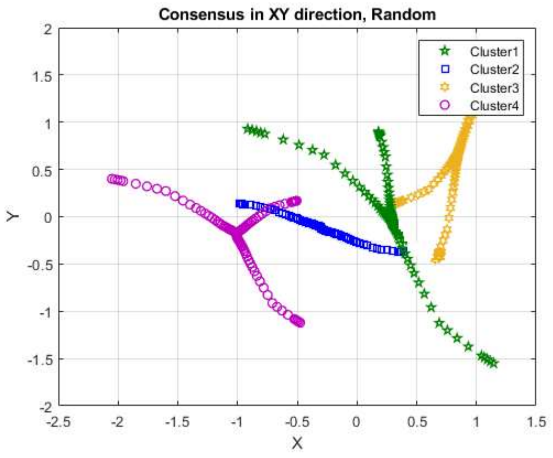
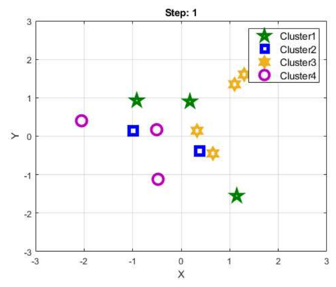
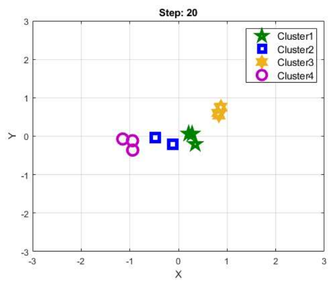


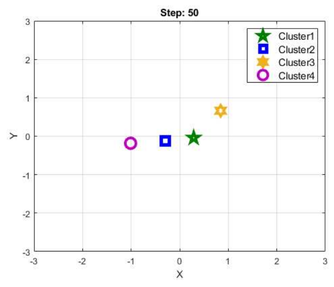
Figure 3.3: The cluster-based movement trajectory of 12 agents in four clusters on $xy-$direction.



(a) Simulation at step 1.



(b) Simulation at step 20.



(c) Simulation at step 50.

Figure 3.4: A simulation of 12 agents' cluster-based motion at three various steps in 2-D space.

Table 3.3: Initial and target positions of all 12 agents for four different clusters in 2-D space (global goal).

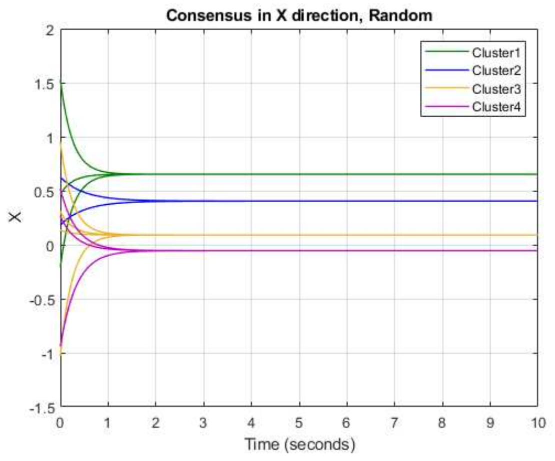
Agent / Cluster	1st Agent	2nd Agent	3rd Agent	4th Agent	Target Position
Cluster 1	$(-0.8236, -0.8314)$	$(-1.5771, -0.9792)$	$(0.5080, -1.1564)$	—	$(-0.3135, -0.3286)$
Cluster 2	$(0.2820, -0.5336)$	$(0.0335, -2.0026)$	—	—	$(-0.3135, -0.3286)$
Cluster 3	$(-1.3337, 0.9642)$	$(1.1275, 0.5201)$	$(0.3502, -0.0200)$	$(-0.2991, -0.0348)$	$(-0.3135, -0.3286)$
Cluster 4	$(0.0229, -0.7982)$	$(-0.2620, 1.0187)$	$(-1.7502, -0.1332)$	—	$(-0.3135, -0.3286)$

Table 3.4: Initial and target positions of all 12 agents for four different clusters in 3-D space (global goal).

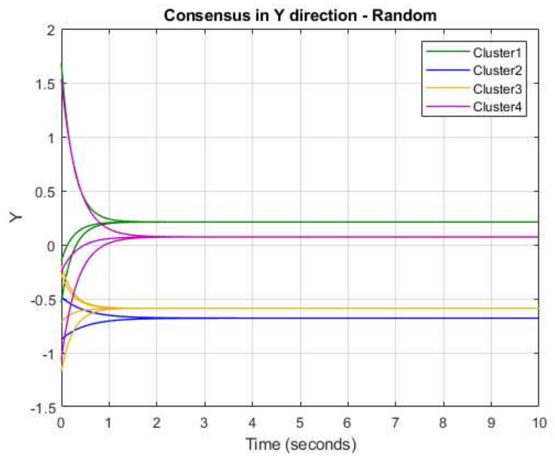
Agent / Cluster	1st Agent	2nd Agent	3rd Agent	4th Agent	Target Position
Cluster 1	$(-0.8657, -0.4140, 1.7463)$	$(-1.0431, -0.4383, 0.1554)$	$(-0.2701, 2.0034, -1.2371)$	—	$(-0.1011, 0.4564, -0.2463)$
Cluster 2	$(-0.4381, 0.9510, -2.1935)$	$(-0.4087, -0.4320, -0.3334)$	—	—	$(-0.1011, 0.4564, -0.2463)$
Cluster 3	$(0.9835, 0.6489, 0.7135)$	$(-0.2977, -0.3601, 0.3174)$	$(1.1437, 0.7059, 0.4136)$	$(-0.5316, 1.4158, -0.5771)$	$(-0.1011, 0.4564, -0.2463)$
Cluster 4	$(0.9726, -1.6045, 0.1440)$	$(-0.5223, 1.0289, -1.6387)$	$(0.1766, 1.4580, -0.7601)$	—	$(-0.1011, 0.4564, -0.2463)$

3.3.2 Reaching Position Consensus of Clusters with Various Goals in 3-D Space

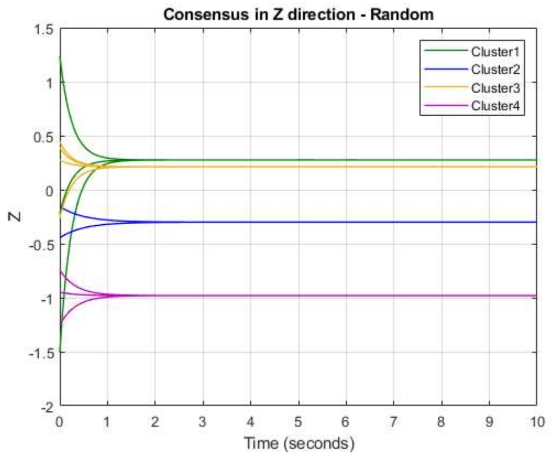
Here, all agents of different clusters reach consensus in a three-dimensional (3-D) space. The steps of doing this experiment are similar to the efforts of Subsection 3.3.1. The only difference is the use of (x,y,z) space instead of (x,y) space. Table 3.2 demonstrates that the agents of each cluster converge to a specific target position. In this regard, all three agents of $Cluster_1$ converge to the point of $(0.6526, 0.2104, 0.2756)$. Furthermore, the agents of $Cluster_2$, and $Cluster_3$ converge to $(0.4042, -0.6798, -0.3003)$, and $(0.0904, -0.5881, 0.2126)$, respectively. Plus, the agents of $Cluster_4$ converge to the target position of $(-0.0549, 0.0723, -0.9799)$. Figures 3.5 and 3.6 illustrate the discussed convergence in 3-D space. Figure 3.5 shows 12 agents' cluster-based convergence of four different clusters on $x-$direction, $y-$direction, and $z-$direction with random initial x, y, and z during 10 seconds. Moreover, Figure 3.6 shows 12 agents' cluster-based movement trajectory and motion simulation in four different clusters on $xyz-$direction.



(a) Consensus on $x-$direction.

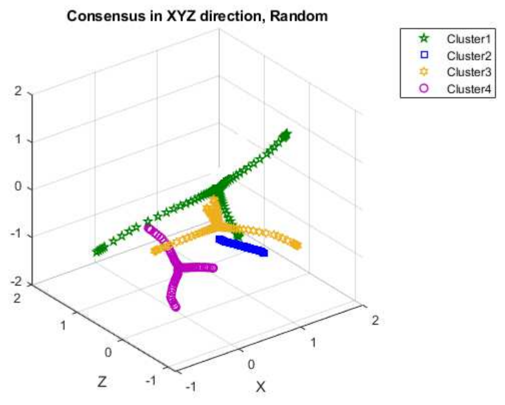


(b) Consensus on $y-$direction.

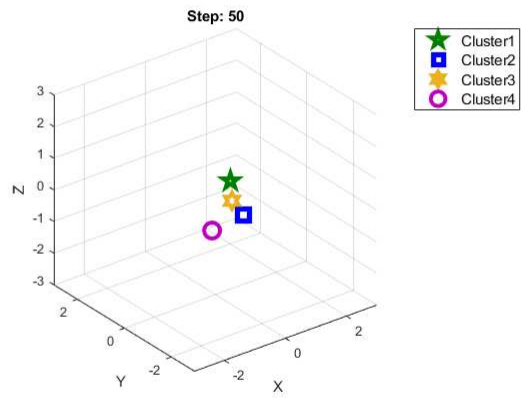


(c) Consensus on $z-$direction.

Figure 3.5: 12 agents of four different clusters reach consensus on $x-$direction, $y-$direction, and $z-$direction with random initial x, y, and z during 10 seconds.

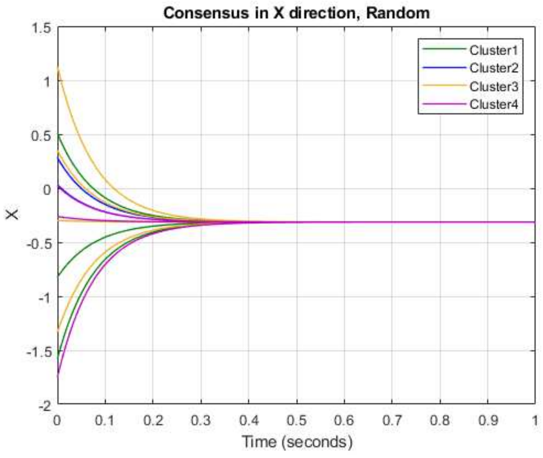


(a) The cluster-based xyz movement trajectory.

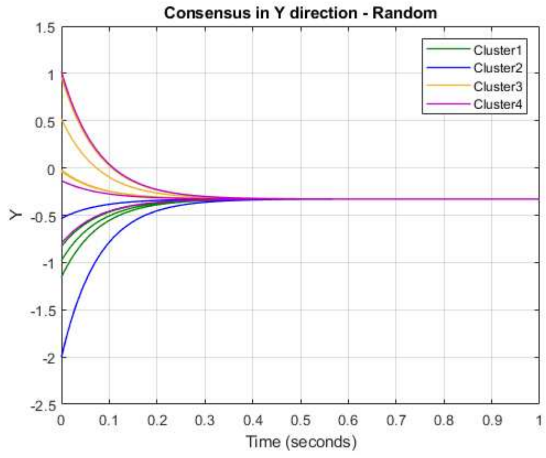


(b) Simulation at step 50 in 3-D space.

Figure 3.6: 12 agents' cluster-based movement trajectory and motion simulation in four clusters on $xyz-$direction.



(a) Consensus on $x-$direction.



(b) Consensus on $y-$direction.

Figure 3.7: 12 agents reach consensus on $x-$direction and $y-$direction with random initial x and y during one second.

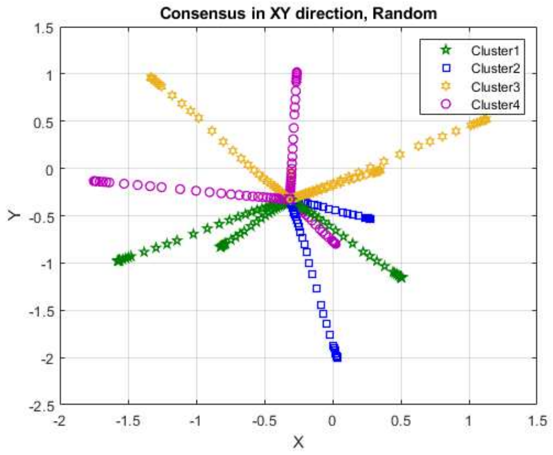
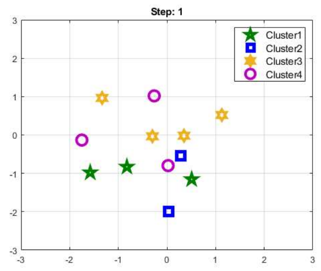
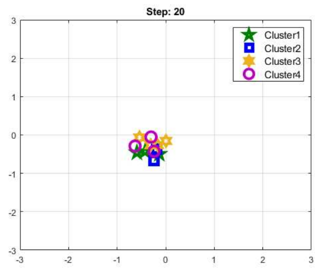


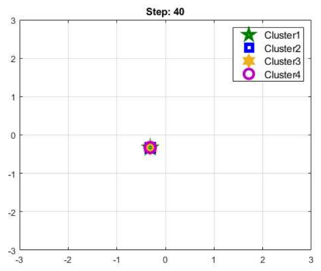
Figure 3.8: The movement trajectory of 12 agents on $xy-$direction.



(a) Simulation at step 1.



(b) Simulation at step 20.



(c) Simulation at step 40.

Figure 3.9: A simulation of 12 agents' motion at three various steps in 2-D space.

3.3.3 Reaching Position Consensus of Clusters with a Global Goal in 2-D Space

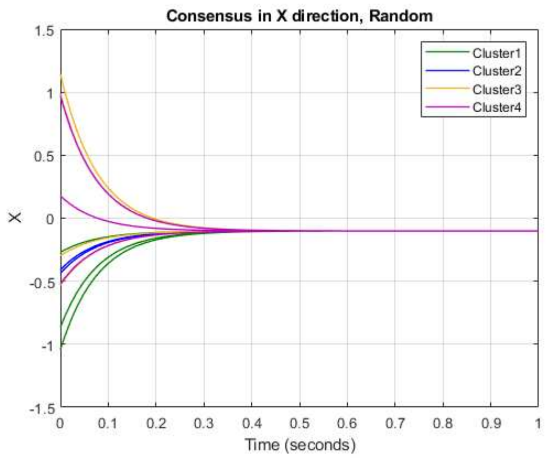
In this subsection all agents are communicated with each others and it is assumed that the entire MAS is a complete graph. Therefore, the graph Laplacian of MAS (including four different clusters) is defined as below:

$$\underset{12\times12}{L} = \underset{12\times12}{D} - \underset{12\times12}{A}$$

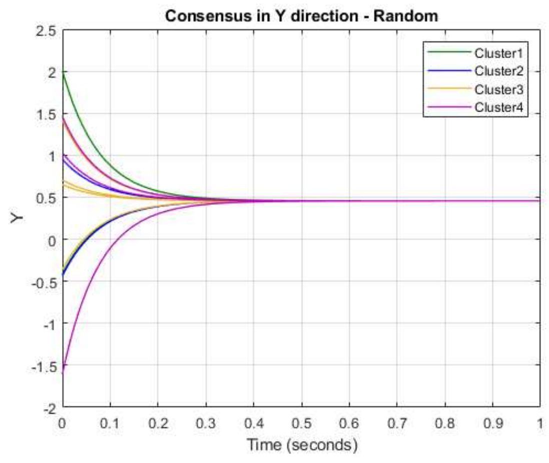
Using (3.19), the set of eigenvalues of Laplacian matrix $\underset{12\times12}{L}$ is obtained as follows:

$$S^{\lambda} = \{0, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12\}.$$

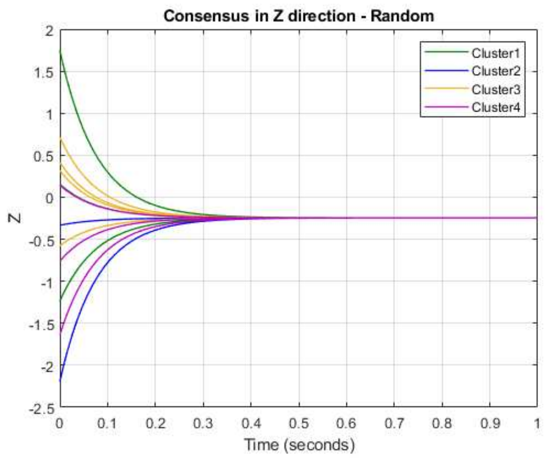
Hence, reaching a consensus is possible for the agents of the MAS. In this scenario, the initial position of each agent is randomly selected. The goal is to reach a common position (consensus on position) for all agents. In this regard, the initial positions of all agents are as Table 3.3. At each time step, agents depending on their current state (current position), make a forward movement (action) to reach the next state (next position), and consequently attain the position consensus. Table 3.3 demonstrates that the agents of all clusters converge to a common target position. In this regard, all agents converge to the point of $(-0.3135, -0.3286)$. Reaching the consensus on $x-$ and $y-$directions (position vs. time) are demonstrated in Figures 3.7a and 3.7b, respectively. All agents have reached a position consensus in the 0.4^{th} seconds. In Figure 3.8 the movement trajectory of all agents is illustrated in both $x-$ and $y-$directions (X vs. Y). Furthermore, the movement of all agents is simulated for 40 steps. Three various steps of this simulation are demonstrated in Figure 3.9.



(a) Consensus on $x-$direction.

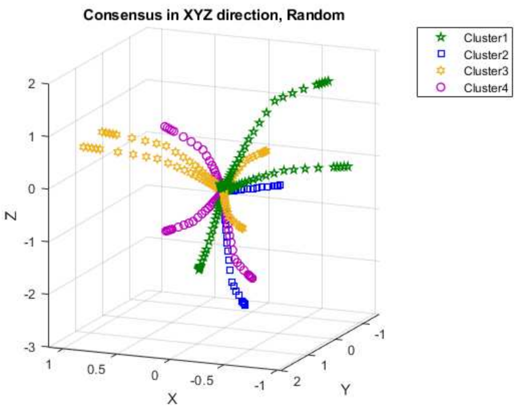


(b) Consensus on $y-$direction.

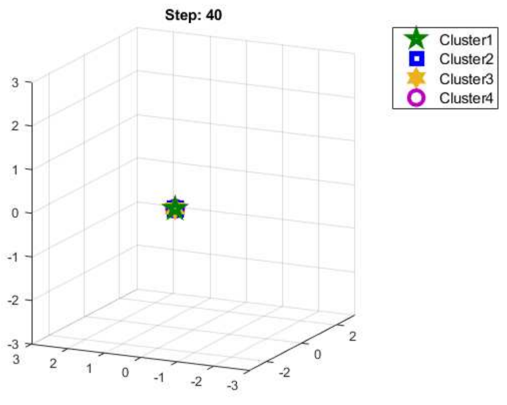


(c) Consensus on $z-$direction.

Figure 3.10: 12 agents reach consensus on $x-$direction, $y-$direction, and $z-$direction with random initial x, y, and z during one second.



(a) The xyz movement trajectory.



(b) Simulation at step 40 in 3-D space.

Figure 3.11: 12 agents' movement trajectory and motion simulation on $xyz-$direction.

3.3.4 Reaching Position Consensus of Clusters with a Global Goal in 3-D Space

All 12 agents reach consensus in a three-dimensional (3-D) space in this experiment. The stages of doing this experiment are identical to the steps of Subsection 3.3.3. The only dissimilarity is using (x, y, z) space instead of (x, y) space. Table 3.4 demonstrates that all agents converge to a global target position. In this regard, all agents converge to the point of $(-0.1011, 0.4564, -0.2463)$. Figures 3.10 and 3.11 illustrate the discussed convergence in the 3-D space. Figure 3.10 shows 12 agents convergence on $x-$direction, $y-$direction, and $z-$direction with random initial x, y, and z during one second. Moreover, Figure 3.11 shows 12 agents' movement trajectory and motion simulation on $xyz-$direction.

It is worth mentioning that for developing and simulating this chapter's method, MATLAB Programming Language is used.

3.4 Conclusions

This chapter considered the average position consensus of cluster-based, heterogeneous MAS using non-learning methods. We presented the discussed average position consensus in two scenarios for 2-D and 3-D spaces. First, the average position consensus was performed when each cluster's goal differed from the other groups. Second, the average position consensus was achieved when all clusters' aim was global. The numerical and simulation results show the successful position consensus for the proposed cluster-based, heterogeneous MAS in 2-D and 3-D spaces.

Chapter 4

Position Consensus of Multi-agent Reinforcement Learning Systems: The Effect of Immediate Rewards

4.1 Introduction

This chapter studies the consensus problem of a leaderless, homogeneous, MARL system using actor-critic algorithms with and without malicious agents. The goal of each agent is to reach the consensus position with the maximum cumulative reward. Although the reward function converges in both scenarios, in the absence of the malicious agent, the cumulative reward is higher than with the malicious agent present. We consider here various immediate reward functions. First, we study the immediate reward function based on Manhattan distance. In addition to proposing three different immediate reward functions based on Euclidean, *n*-norm, and Chebyshev distances, we have rigorously shown which method has a better performance based on a cumulative reward for each agent and the entire team of agents. Finally, we present a combination of various immediate reward functions that yields a higher cumulative reward for each agent and the team of agents. By increasing the agents' cumulative reward using the combined immediate reward function, we have demonstrated that the cumulative team reward in the presence of a malicious agent is comparable with the cumulative team reward in the absence of the malicious agent. The claims have been proven theoretically, and the simulation confirms theoretical findings.

4.2 Background

For decision-making, each agent applies the information received from the environment. The finite MDP is considered to represent the dynamics of the environment for decision-making in choosing the best action. An MDP for a MARL system can be defined by a 5-tuple $M = \langle S, A, p, R, \gamma \rangle$, where $S = S_1 \times S_2 \times ... \times S_N$ is a finite set and Cartesian product of environmental states, $A = A_1 \times A_2 \times ... \times A_N$ is a finite joint action set for all agents so that $A_i = a_1 \times a_2 \times ... \times a_K$, $i = 1, 2, ..., N$, is a set of actions of each agent, $p : S \times A \times S^{'} \to [0,1]$, describes the environment's dynamics is the state-transition probability function that agents starts in state S, takes action A, and ends in state $S^{'}$. Further, $R : S \times A \times S^{'} \to \mathrm{I\!R}^n$ is a reward function. For i^{th} agent $R^i_{t+1} = \mathbb{E}\left[r^i_{t+1} | s_t = s, a^i_t = a\right]$, where r^i_{t+1} indicates the immediate reward, s_t shows the state, and a^i_t is action at time t. In an MDP for a MARL system, the cumulative reward is expected to be maximized for all agents, as well as the team of agents [205]. The trade-off between an immediate reward and potential future reward is determined by the discount factor $\gamma \in [0,1)$.

The MAS is considered as the graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of all vertices (agents), and $\mathscr{E} \subseteq \{(i,j) | i \in \mathscr{V}, j \in \mathscr{V}\}$ is the set of all edges (communication links between agents). The agents i and j are neighbors if and only if $(i,j) \in \mathscr{E}$.

4.3 Methodology

This chapter investigates increasing the agents' cumulative reward for two scenarios: with and without malicious agents in the MARL system.

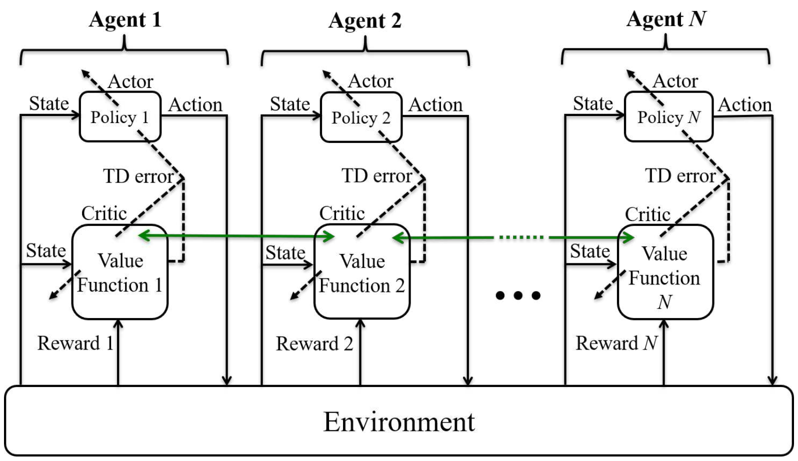


Figure 4.1: Multi-agent actor-critic architecture with N agents. The green arrows indicate transferring correct data between neighboring agents.

4.3.1 Without Malicious Agents

The goal of each agent is to reach the position consensus in the environment with the maximum cumulative reward. The considered environment in this chapter is a grid world. We consider a MAS with corresponding actor-critic architecture [128]. An actor-critic architecture is assigned to each agent in the MAS, where the neighboring agents communicate with each other via the critic unit as illustrated in Figure 4.1. Each agent is trained to learn the local policy utilizing decentralized learning.

The set of independent local policies for N agents is described as $Policy = \{\pi_1, \pi_2, ..., \pi_N\}$. At time $t = 0$, all agents are assigned the initial state s_0. The actor unit of i^{th} agent uses the policy function $\pi_i(a_0^i|s_0)$ to perform the action a_0^i related to the initial state s_0. At time $t+1$, all agents receive state s_{t+1}, as well as a local immediate reward r_{t+1}^i from the environment according to the action a_t^i they performed at time t. Each agent keeps the immediate reward information r_{t+1}^i; however, they are permitted to estimate the immediate reward of the network. Based on the reward r_{t+1}^i and state s_{t+1}, the critic unit of i^{th} agent examines whether the actor unit has taken the appropriate action to improve the agent's selection in the following steps. For this purpose, at time $t+1$, the critic unit estimates the reward $\hat{r}_{t+1}^i$ and compares it with the environmental received reward r_{t+1}^i. The estimation of $\hat{r}_{t+1}^i$ is done using a four-layer NN including input layer (environmental received states are fed to this layer), two hidden dense layers, and a dense output layer to return the estimated reward $\hat{r}^i_{t+1}$.

The comparison between the estimated reward $\hat{r}^i_{t+1}$ and the environmental received reward r^i_{t+1} is carried out using the temporal difference (TD) error. The higher value of the TD error means the greater difference between the actual reward r^i_{t+1} and expected reward $\hat{r}^i_{t+1}$. The TD error for the i^{th} agent, δ^i_t, is given by

$$\delta^i_t = R^i_{t+1} + \gamma V^i_t(s_{t+1}) - V^i_t(s_t), \tag{4.1}$$

where $V^i_t(s_t)$ is the critic value function at time t defined by

$$V^i_t(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^i_{t+1} | s_t = s\right], \tag{4.2}$$

where s_t is state of i^{th} agent at time t which is defined as the current position (x^i, y^i) in 2-D space. Using (4.1) and (4.2), the TD error method yields

$$V^i_{t+1}(s_t) = V^i_t(s_t) + \alpha \delta^i_t, \tag{4.3}$$

where α is the learning rate. The TD error value of i^{th} agent is sent to the actor unit of the current agent to improve the following action selection, as well as to the critic units of the neighboring agents through the communication links using a consensus protocol.

In the utilized algorithm, the consensus step is as follows:

$$\begin{cases} \lambda^i_{t+1} \leftarrow \sum_{j \in \mathcal{N}} w_t(i,j) \cdot \tilde{\lambda}^j_t, \\ v^i_{t+1} \leftarrow \sum_{j \in \mathcal{N}} w_t(i,j) \cdot \tilde{v}^j_t, \end{cases} \tag{4.4}$$

where λ and $\tilde{\lambda}$ are the actual and estimated multi-agent reward function parameters, respectively. Furthermore, v and $\tilde{v}$ are the actual and estimated multi-agent value function parameters, respectively. According to [132], and [128] the initialization of the parameters is performed for λ, $\tilde{\lambda}$, v, and $\tilde{v}$ at time $t = 0$ for all N agents. The discussed parameters should be updated and added to the list of previous values at $t+1$. Moreover, v^i of each agent describes the network value function approximation $V_t^i(s_t; v_t^i)$. Hence,

$$\tilde{\lambda}_t^i \leftarrow \lambda_t^i + \alpha_{v,t}\left(r_{t+1}^i - \hat{r}_{t+1}\left(\lambda_t^i\right)\right)\nabla_\lambda \hat{r}_{t+1}\left(\lambda_t^i\right) \tag{4.5}$$

$$\tilde{v}_t^i \leftarrow v_t^i + \alpha_{v,t}\delta_t^i \nabla_v V_t^i\left(s_t; v_t^i\right). \tag{4.6}$$

Besides, $\mathcal{N}$ is the set of neighbors of i^{th} agent, and $W_t = [w_t(i,j)]_{N\times N}$ is Metropolis weight matrix specified by

$$W_t = \begin{cases} \dfrac{1}{1+\max\{d_t(i), d_t(j)\}} & \text{if } (i,j) \in \mathcal{E}, \\ 1 - \sum\limits_{(i,k)\in\mathcal{E}} W_t(i,k) & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

with $d_t(i)$ and $d_t(j)$ being the degree of agents i and j, respectively. Therefore, the weight on the message transferred from agent i to agent j at time t is $w_t(i,j)$. The consensus step (4.4) must be done by all N agents in the MARL system to reach the position consensus. Updating the reward function parameter λ^i and value function parameter v^i enables the i^{th} agent to update its policy function $\pi_i(a_t^i|s_t)$. Note that the structure of weight matrix W_t depends on the communication graph topology [128].

4.3.2 With Malicious Agents

One of the problems that can occur with any MAS is an incompatibility of one or more agents with the other agents. These types of agents, termed malicious agents, may be internally disturbed and can have a negative effect on MAS performance. In this chapter, the malicious agent does not apply the consensus updates and skips the consensus step of (4.4), which results in an unbalanced consensus throughout the entire MAS [132]. An actor-critic MARL system with a malicious agent

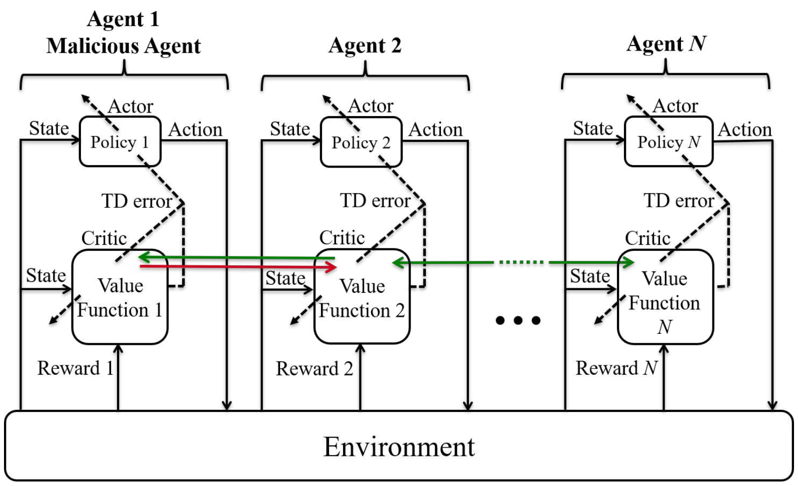


Figure 4.2: Multi-agent actor-critic architecture with N agents. The green arrows indicate transferring correct data between neighboring agents, and the red arrow represents transmitting inaccurate data from malicious agent to neighboring agents.

is illustrated in Figure 4.2. At the time t, the malicious agent receives correct data from the critic units of neighboring agents, including TD error. However, the malicious agent sends inaccurate information to neighboring agents' critic units via communication links or performs adverse actions, causing this agent to maximize its cumulative reward. Simultaneously, the cumulative rewards of neighboring agents are reduced due to improper information they receive from the malicious agent. MAS's cumulative team reward is reduced compared to the situation where there is no malicious agent in the system [132]. In the following, the immediate reward function and its effect on the system performance are analyzed.

4.3.3 Reward Functions

Choosing an appropriate reward function is a significant challenge in RL algorithms. There is no specific rule to select or define an immediate reward function. In general, one should select the immediate reward function based on the RL system's application. We consider five various immediate reward functions, based on multiple distance metrics, to reach the position consensus. For all the following distance metrics and immediate reward functions, (x, y) and (x_{des}, y_{des}) are the current position and the desired position, respectively. Moreover, (x^i, y^i) and (x^i_{des}, y^i_{des}) are the current position and the desired position of the i^{th} agent, sequentially.

4.3.3.1 Manhattan Immediate Reward

Each agent's immediate reward in 2-D space is determined based on the Manhattan distance: $M_d = |x - x_{des}| + |y - y_{des}|$. The Manhattan immediate reward function for i^{th} agent is given by [132]:

$$Mr^i_{t+1} = -\left|x^i - x^i_{des}\right| - \left|y^i - y^i_{des}\right|. \tag{4.8}$$

4.3.3.2 Euclidean Immediate Reward

Based on the Euclidean distance E_d, we define the Euclidean immediate reward function for the i^{th} agent in 2-D space

$$Er^i_{t+1} = -\left(\left|x^i - x^i_{des}\right|^2 + \left|y^i - y^i_{des}\right|^2\right)^{1/2}. \tag{4.9}$$

4.3.3.3 *n*-norm Immediate Reward

Using the n-norm metric $N_d = (|x - x_{des}|^n + |y - y_{des}|^n)^{1/n}$, the immediate reward function for the i^{th} agent in 2-D space is given by

$$Nr^i_{t+1} = -\left(\left|x^i - x^i_{des}\right|^n + \left|y^i - y^i_{des}\right|^n\right)^{1/n}, \tag{4.10}$$

where $n \geq 3$.

4.3.3.4 Chebyshev Immediate Reward

Utilizing the Chebyshev distance metric $\check{C}_d = \max(|x - x_{des}|, |y - y_{des}|)$, the Chebyshev immediate reward function for i^{th} agent in 2-D space is given by

$$\check{C}r^i_{t+1} = \max\left(-\left|x^i - x^i_{des}\right|, -\left|y^i - y^i_{des}\right|\right). \tag{4.11}$$

4.3.3.5 Combined Immediate Reward

Based on the immediate reward functions (4.8)-(4.11), the combined immediate reward function for i^{th} agent in 2-D space is given by

$$Cr_{t+1}^{i} = \max\left(Mr_{t+1}^{i}, Er_{t+1}^{i}, Nr_{t+1}^{i}, \check{C}r_{t+1}^{i}\right). \tag{4.12}$$

Equation (4.12), in each episode and for all agents, calculates the various immediate rewards of (4.8)-(4.11) and selects the maximum reward based on the returned values. This method uses other immediate reward functions to get the largest cumulative reward during each episode.

Theorem 4.1 *Let Mr_{t+1}^{i} and Er_{t+1}^{i} be Manhattan and Euclidean immediate reward functions for i^{th} agent in 2-D space, then the Euclidean cumulative team reward is greater than or equal to the Manhattan cumulative team reward for N agents in 2-D space.*

Proof 4.1 *For i^{th} agent in 2-D space, $|\Delta x| = |x^{i} - x_{des}^{i}|$, and $|\Delta y| = |y^{i} - y_{des}^{i}|$. According to $|\Delta x|^2 + |\Delta y|^2 \leq (|\Delta x| + |\Delta y|)^2$ and by considering the positive roots of both sides of inequality, the following is valid*

$$-\left|x^{i} - x_{des}^{i}\right| - \left|y^{i} - y_{des}^{i}\right| \leq -\left(\left|x^{i} - x_{des}^{i}\right|^2 + \left|y^{i} - y_{des}^{i}\right|^2\right)^{1/2}. \tag{4.13}$$

From (4.13), it is concluded that the Manhattan immediate reward is less than or equal to the Euclidean immediate reward. Therefore, from $Mr_{t+1}^{i} \leq Er_{t+1}^{i}$ it is obvious that

$$\mathbb{E}\left[Mr_{t+1}^{i} | s_t = s, a_t^{i} = a\right] \leq \mathbb{E}\left[Er_{t+1}^{i} | s_t = s, a_t^{i} = a\right] \tag{4.14}$$

$$MR_{t+1}^{i} \leq ER_{t+1}^{i}, \tag{4.15}$$

where MR_{t+1}^{i} and ER_{t+1}^{i} are Manhattan and Euclidean cumulative rewards for i^{th} agent in 2-D *space, respectively. Using (4.15) we have*

$$\frac{1}{N}\sum_{i=1}^{N} MR_{t+1}^{i} \leq \frac{1}{N}\sum_{i=1}^{N} ER_{t+1}^{i}. \tag{4.16}$$

Hence, the Euclidean cumulative team reward is greater than or equal to the Manhattan cumulative team reward for N agents in 2-D space.

Remark 4.1 *Since Manhattan and Euclidean distances are called* 1*-norm and* 2*-norm distances, respectively, then the proof of Theorem 4.1 can be expanded to show that the n-norm cumulative team reward ($n \geq 3$) is greater than or equal to the Euclidean cumulative team reward for N agents in 2-D space.*

Theorem 4.2 *Let Er_{t+1}^{i} and $\check{C}r_{t+1}^{i}$ be Euclidean and Chebyshev immediate reward functions for i^{th} agent in 2-D space, then the Chebyshev cumulative team reward is greater than or equal to the Euclidean cumulative team reward for N agents in 2-D space.*

Proof 4.2 *Based on triangle inequality, the product of Chebyshev distance is always less than or equal to the outcome of Euclidean distance ($\check{C}_d \leq E_d$). Hence, the following is valid*

$$-\left(|\Delta x|^2 + |\Delta y|^2\right)^{1/2} \leq -\max\left(|\Delta x|, |\Delta y|\right). \tag{4.17}$$

We know that

$$-\max\left(|\Delta x|, |\Delta y|\right) \leq \max\left(-|\Delta x|, -|\Delta y|\right). \tag{4.18}$$

Using (4.17) and (4.18) yields

$$-\left(\left|x^i - x_{des}^i\right|^2 + \left|y^i - y_{des}^i\right|^2\right)^{1/2} \leq \max\left(-\left|x^i - x_{des}^i\right|, -\left|y^i - y_{des}^i\right|\right). \tag{4.19}$$

From (4.19), it is derived that $Er^i_{t+1} \leq \check{C}r^i_{t+1}$. Afterward,

$$\mathbb{E}\left[Er^i_{t+1}|s_t = s, a^i_t = a\right] \leq \mathbb{E}\left[\check{C}r^i_{t+1}|s_t = s, a^i_t = a\right] \tag{4.20}$$

$$ER^i_{t+1} \leq \check{C}R^i_{t+1}, \tag{4.21}$$

where $\check{C}R^i_{t+1}$ is the Chebyshev cumulative reward for i^{th} agent in 2-D space. Using (4.21) we have

$$\frac{1}{N}\sum_{i=1}^{N} ER^i_{t+1} \leq \frac{1}{N}\sum_{i=1}^{N} \check{C}R^i_{t+1}. \tag{4.22}$$

Therefore, the Chebyshev cumulative team reward is greater than or equal to the Euclidean cumulative team reward for N agents in 2-D space.

Theorem 4.3 *Let Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, $\check{C}r^i_{t+1}$, and Cr^i_{t+1} be Manhattan, Euclidean, n-norm, Chebyshev, and combined immediate reward functions for i^{th} agent in 2-D space, respectively. Then, the combined cumulative team reward for N agents is greater than or equal to the maximum of the Manhattan, Euclidean, n-norm, and Chebyshev cumulative team rewards for the same N agents in 2-D space during each episode.*

Proof 4.3 *From (4.12) it follows*

$$\mathbb{E}\left[Cr^i_{t+1}|s_t = s, a^i_t = a\right] \geq \mathbb{E}\left[Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, \check{C}r^i_{t+1}|s_t = s, a^i_t = a\right]. \tag{4.23}$$

Given that Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, and $\check{C}r^i_{t+1}$ are independent functions, also by taking the maximum function from both sides of inequality, one has

$$\begin{aligned}\mathbb{E}\left[Cr^i_{t+1}|s_t = s, a^i_t = a\right] \geq \max\Big(&\mathbb{E}\left[Mr^i_{t+1}|s_t = s, a^i_t = a\right], \mathbb{E}\left[Er^i_{t+1}|s_t = s, a^i_t = a\right], \\ &\mathbb{E}\left[Nr^i_{t+1}|s_t = s, a^i_t = a\right], \mathbb{E}\left[\check{C}r^i_{t+1}|s_t = s, a^i_t = a\right]\Big).\end{aligned} \tag{4.24}$$

As a consequence, we have

$$CR^i_{t+1} \geq \max\left(MR^i_{t+1}, ER^i_{t+1}, NR^i_{t+1}, \check{C}R^i_{t+1}\right), \tag{4.25}$$

where NR^i_{t+1} and CR^i_{t+1} are n-norm and combined cumulative rewards, respectively. From (4.25), it follows that the combined cumulative reward for i^{th} agent is greater than or equal to the maximum of Manhattan, Euclidean, n-norm, and Chebyshev cumulative rewards for the same agent in 2-D space during each episode. Therefore,

$$\frac{1}{N}\sum_{i=1}^{N} CR^i_{t+1} \geq \sum_{i=1}^{N} \max\left(\frac{1}{N}MR^i_{t+1}, \frac{1}{N}ER^i_{t+1}, \frac{1}{N}NR^i_{t+1}, \frac{1}{N}\check{C}R^i_{t+1}\right). \tag{4.26}$$

We know that

$$\begin{aligned}&\sum_{i=1}^{N} \max\left(\frac{1}{N}MR^i_{t+1}, \frac{1}{N}ER^i_{t+1}, \frac{1}{N}NR^i_{t+1}, \frac{1}{N}\check{C}R^i_{t+1}\right) \geq \\ &\max \sum_{i=1}^{N}\left(\frac{1}{N}MR^i_{t+1}, \frac{1}{N}ER^i_{t+1}, \frac{1}{N}NR^i_{t+1}, \frac{1}{N}\check{C}R^i_{t+1}\right).\end{aligned} \tag{4.27}$$

Using (4.26) and (4.27) yields

$$\frac{1}{N}\sum_{i=1}^{N} CR^i_{t+1} \geq \max\left(\frac{1}{N}\sum_{i=1}^{N}MR^i_{t+1}, \frac{1}{N}\sum_{i=1}^{N}ER^i_{t+1}, \frac{1}{N}\sum_{i=1}^{N}NR^i_{t+1}, \frac{1}{N}\sum_{i=1}^{N}\check{C}R^i_{t+1}\right). \tag{4.28}$$

Hence, at time t, the combined cumulative team reward for N agents is greater than or equal to the maximum of the Manhattan, Euclidean, n-norm, and Chebyshev cumulative team rewards for the same N agents in 2-D space during each episode.

Theorem 4.4 *Let Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, $\check{C}r^i_{t+1}$, and Cr^i_{t+1} be Manhattan, Euclidean, n-norm, Chebyshev, and combined immediate reward functions for i^{th} agent in 2-D space, respectively. Then, the combined critic value for i^{th} agent is greater than or equal to the maximum of Manhattan, Euclidean, n-norm, and Chebyshev critic values for the same agent in 2-D space during each episode.*

Proof 4.4 *Since $\gamma \in [0,1)$ and $t \in [0,\infty)$, it is concluded that $\gamma^t \in [0,1]$. Therefore, from (4.12) we have*

$$\sum_{t=0}^{\infty} \gamma^t Cr_{t+1}^i = \sum_{t=0}^{\infty} \max\left(\gamma^t Mr_{t+1}^i, \gamma^t Er_{t+1}^i, \gamma^t Nr_{t+1}^i, \gamma^t \check{C}r_{t+1}^i\right). \tag{4.29}$$

We know that

$$\sum_{t=0}^{\infty} \max\left(\gamma^t Mr_{t+1}^i, \gamma^t Er_{t+1}^i, \gamma^t Nr_{t+1}^i, \gamma^t \check{C}r_{t+1}^i\right) \geq \max \sum_{t=0}^{\infty} \left(\gamma^t Mr_{t+1}^i, \gamma^t Er_{t+1}^i, \gamma^t Nr_{t+1}^i, \gamma^t \check{C}r_{t+1}^i\right). \tag{4.30}$$

After simplifying, using (4.29) and (4.30) yields

$$\sum_{t=0}^{\infty} \gamma^t Cr_{t+1}^i \geq \sum_{t=0}^{\infty} \left(\gamma^t Mr_{t+1}^i, \gamma^t Er_{t+1}^i, \gamma^t Nr_{t+1}^i, \gamma^t \check{C}r_{t+1}^i\right). \tag{4.31}$$

By taking expectation with respect to the state from both sides of inequality, the following is achieved

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Cr_{t+1}^i | s_t = s\right] \geq \mathbb{E}\left[\left(\sum_{t=0}^{\infty} \gamma^t Mr_{t+1}^i, \sum_{t=0}^{\infty} \gamma^t Er_{t+1}^i, \sum_{t=0}^{\infty} \gamma^t Nr_{t+1}^i, \sum_{t=0}^{\infty} \gamma^t \check{C}r_{t+1}^i\right) \Big| s_t = s\right]. \tag{4.32}$$

Since $\sum_{t=0}^{\infty} \gamma^t Mr_{t+1}^i$, $\sum_{t=0}^{\infty} \gamma^t Er_{t+1}^i$, $\sum_{t=0}^{\infty} \gamma^t Nr_{t+1}^i$, and $\sum_{t=0}^{\infty} \gamma^t \check{C}r_{t+1}^i$ are statistically independent, after simplifying and taking the maximum function from both sides of (4.32), the following is obtained

$$\begin{aligned}\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Cr_{t+1}^i | s_t = s\right] \geq \max\Bigg(&\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Mr_{t+1}^i | s_t = s\right], \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Er_{t+1}^i | s_t = s\right], \\ &\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Nr_{t+1}^i | s_t = s\right], \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \check{C}r_{t+1}^i | s_t = s\right]\Bigg).\end{aligned} \tag{4.33}$$

Therefore,

$$CV_t^i(s_t) \geq \max\left(MV_t^i(s_t), EV_t^i(s_t), NV_t^i(s_t), \check{C}V_t^i(s_t)\right), \tag{4.34}$$

where $MV_t^i(s_t)$, $EV_t^i(s_t)$, $NV_t^i(s_t)$, $\check{C}V_t^i(s_t)$, and $CV_t^i(s_t)$ are Manhattan, Euclidean, n-norm, Chebyshev, and combined critic value functions. Therefore, the combined critic value for i^{th} agent is *greater than or equal to the maximum of Manhattan, Euclidean, n-norm, and Chebyshev critic values for the same agent in 2-D space during each episode.*

4.4 Results and Discussion

This section demonstrates results for consensus control of MAS with and without malicious agents, using the RL decentralized actor-decentralized critic method. To reach the position consensus, a fully connected graph $\mathscr{G}$ is considered, which is illustrated in Figure 4.5. Each actor's internal structure consists of a fully-connected NN architecture for training, including three dense layers with Adam optimizer and categorical cross-entropy loss function. The first and second layers' activation functions are rectified linear unit (ReLU) functions, and the third layer has the softmax activation function.

Similar to the actor, each agent's critic has a three-layer, fully-connected NN structure, including the ReLU activation functions for the first two layers, utilizing Adam optimizer and MSE loss function. The NN structure for training the reward function is similar to the architecture used for training all agents critic. This section's results are derived from MAT-files, obtained by training the NN above for each agent. Each MAT-file is a cell of 200 structures (number of episodes to train); each structure contains state, action, reward, and predicted reward for five agents in 1000 time-steps. In this chapter, evaluating the performance of the utilized RL algorithm is done by considering how much reward each agent and a team of agents receive while acting, and then showing the cumulative reward as a function of the episodes and number of steps.

First, reaching the position consensus on the X-axis is shown in the 1000 time-steps for five agents, using the Manhattan immediate reward function. Then, the average reward during 200 episodes is displayed using five immediate reward functions. Afterward, each agent's average cumulative reward and the average cumulative team reward using different immediate reward functions during 200 episodes with and without a malicious agent are compared. Note that action space consists

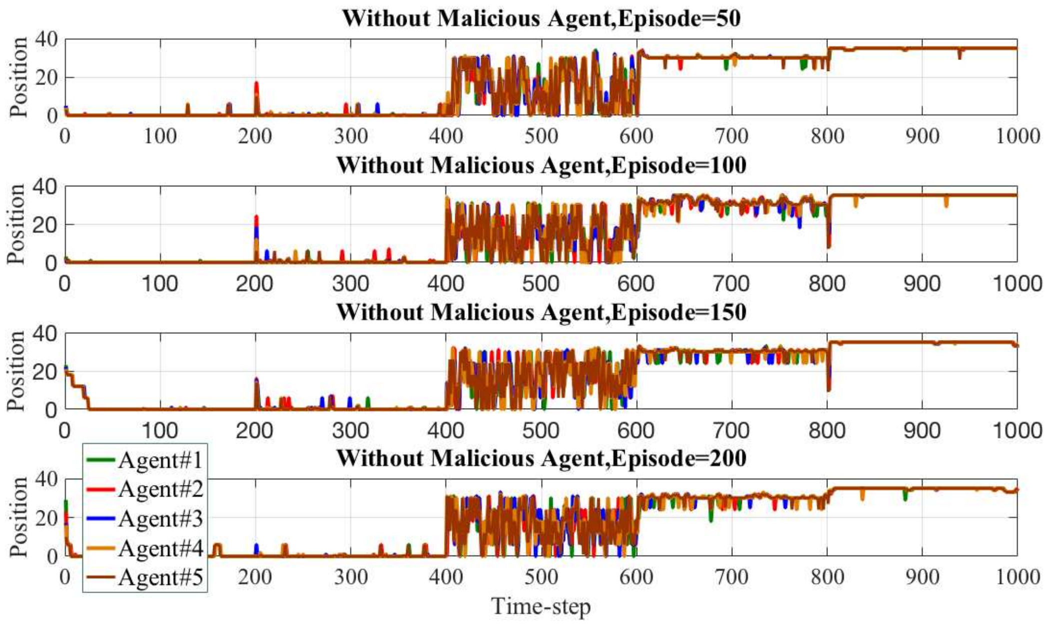


Figure 4.3: The MARL system's performance in reaching the position consensus without a malicious agent at episodes 50, 100, 150, 200 on X-axis.

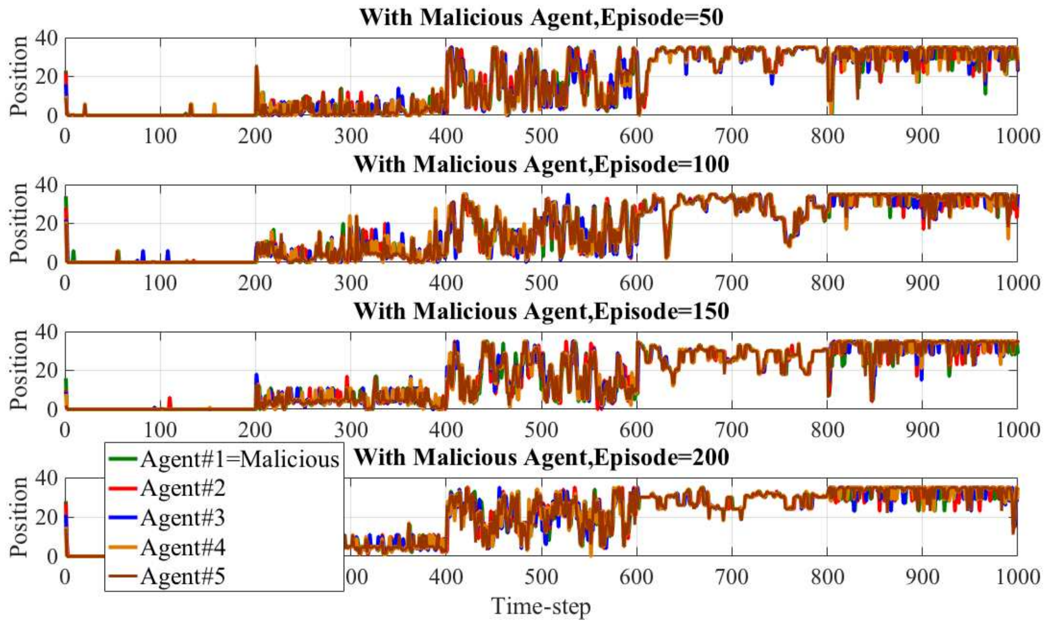
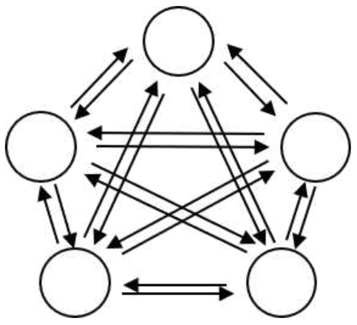
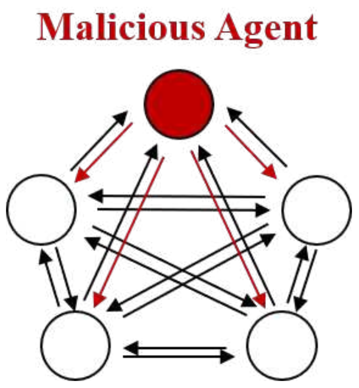


Figure 4.4: The MARL system's performance in reaching the position consensus with a malicious agent at episodes 50, 100, 150, 200 on X-axis.


(a) Topology of a fully connected graph $\mathscr{G}$ without a malicious agent.



(b) Topology of a fully connected graph $\mathscr{G}$ with a malicious agent.

Figure 4.5: A fully connected graph $\mathscr{G}$ is considered as the MARL system, including $N = 5$ nodes. The malicious agent (red circle) refuses to update the parameters in the consensus step.

of five distinctive actions, including waiting and also move to the right, left, up, and down. The actor and critic learning rates are $\alpha = 0.001$ and $\alpha = 0.01$, respectively, and the discount factor is $\gamma = 0.95$.

We have used and extended a part of the code provided in [206] for a part of our implementation. Moreover, the algorithm's execution is done using a system with 3.60 GHz Intel Core i7 $-$ 7700 processor, 16 GB installed RAM, 64$-$bit operating system, and x64$-$based processor.

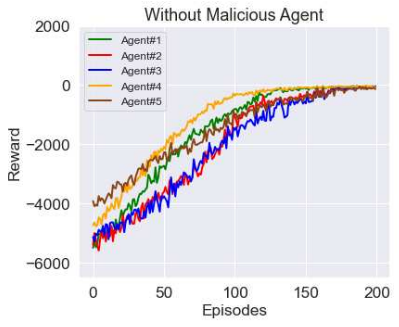
4.4.1 Reaching Consensus

The position consensus of five agents on the X-axis with and without a malicious agent are illustrated in Figures 4.3 and 4.4 at episodes 50, 100, 150, and 200. This consensus is demonstrated during 1000 time-steps using Manhattan immediate reward function. The initial position for i^{th} agent is randomly selected. The desired position for i^{th} agent is $x^i_{des} = 35$. As shown in Figures 4.3 and 4.4, the position convergence of the MARL system in the absence of a malicious agent is superior to the position convergence with a malicious agent's presence during 200 episodes. According to Figure 4.3, the agents' convergence behaviour is observed in the episode 50; however, according to Figure 4.4, this behaviour has not appeared during 200 episodes. The cumulative team reward of the system without malicious agents is greater than the system's cumulative team reward with a malicious agent (Figure 4.6). Accordingly, the MARL system's performance in reaching the consensus without a malicious agent is superior to the network performance with a malicious agent during 200 episodes. Therefore, to improve the network performance in Figure 4.4, the system's cumulative team reward with the malicious agent should increase, which we will examine in the following.

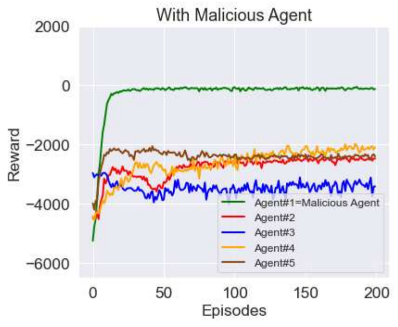
4.4.2 Increasing the Cumulative Reward

When no malicious agents exist in the MARL system, the agents' goal is to maximize the sum of all cumulative rewards. Figure 4.6a shows the reward vs. episodes diagram of five agents without malicious agents. The cumulative reward of all agents reaches the maximum value during 200 episodes. As Figure 4.6a shows, all agents have learned the optimal policy almost equally and have maximized their cumulative reward. We examine the agents' reaction of a MARL system if a malicious agent is detected within the system.

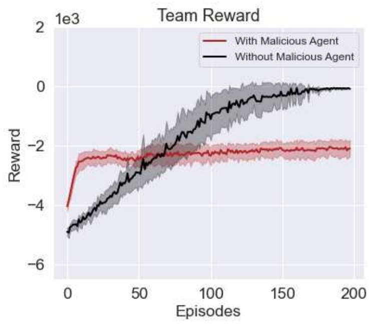
An agent is considered as a malicious agent when it seeks to maximize its own cumulative reward only. The reward vs. episodes diagram of five agents with the malicious agent's presence is illustrated in Figure 4.6b during 200 episodes. Indeed, `Agent#1` is the malicious agent, and its cumulative reward is maximized. The other four agents are not able to maximize their cumulative reward as much as they did in the previous step and cannot learn the optimal policy precisely. However, they enhanced their cumulative reward. Thus, as shown in Figure 4.6c, the cumulative team reward of the MARL system converges without the presence of a malicious agent and is superior to the cumulative team reward of the MARL system with the presence of a malicious agent. The malicious agent has caused the cumulative team reward to converge to -2291.05. All diagrams of Figure 4.6 are obtained using the Manhattan immediate reward function defined in (4.8).



(a) Average reward without malicious agents.

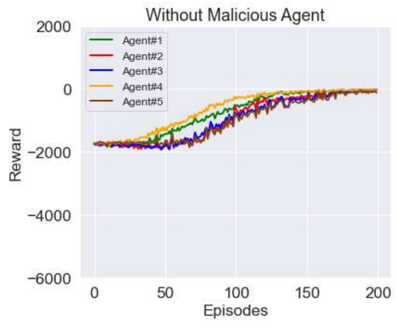


(b) Average reward with a malicious agent.

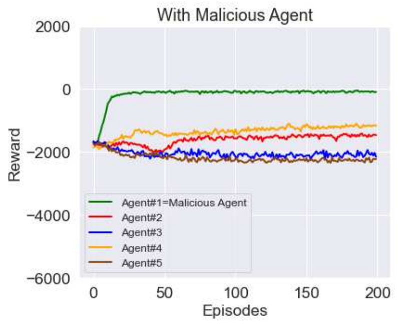


(c) Team reward with and without a malicious agent.

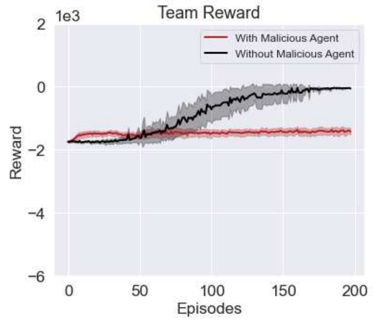
Figure 4.6: Reward convergence using the Manhattan immediate reward function during 200 episodes for $N = 5$ agents.



(a) Average reward without malicious agents.

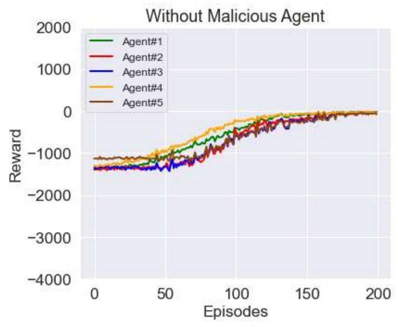


(b) Average reward with a malicious agent.

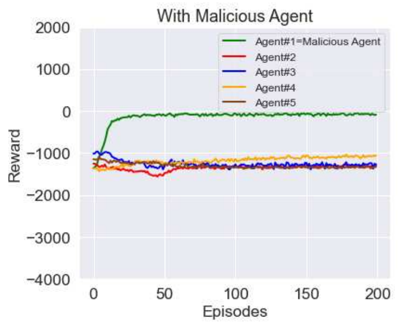


(c) Team reward with and without a malicious agent.

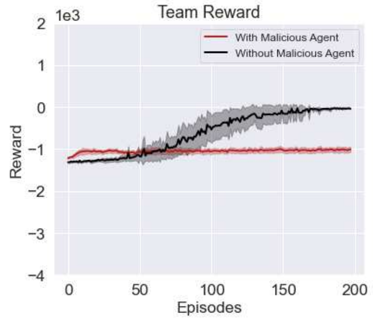
Figure 4.7: Reward convergence using the Euclidean immediate reward function during 200 episodes for $N = 5$ agents.



(a) Average reward without malicious agents.

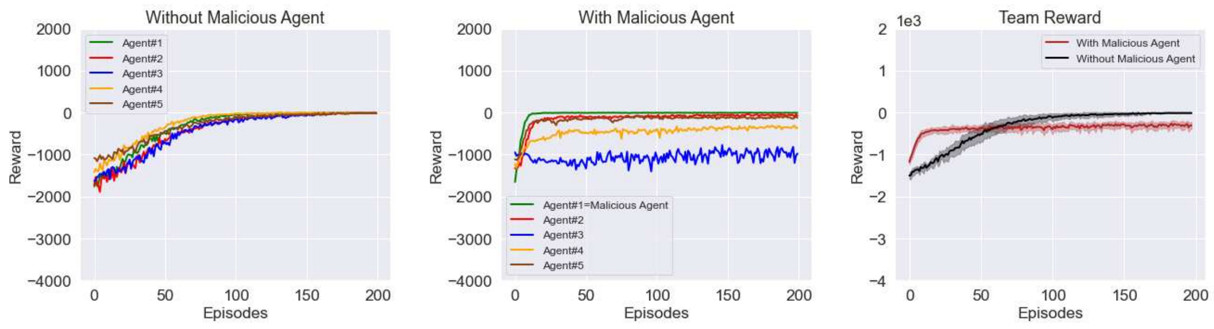


(b) Average reward with a malicious agent.



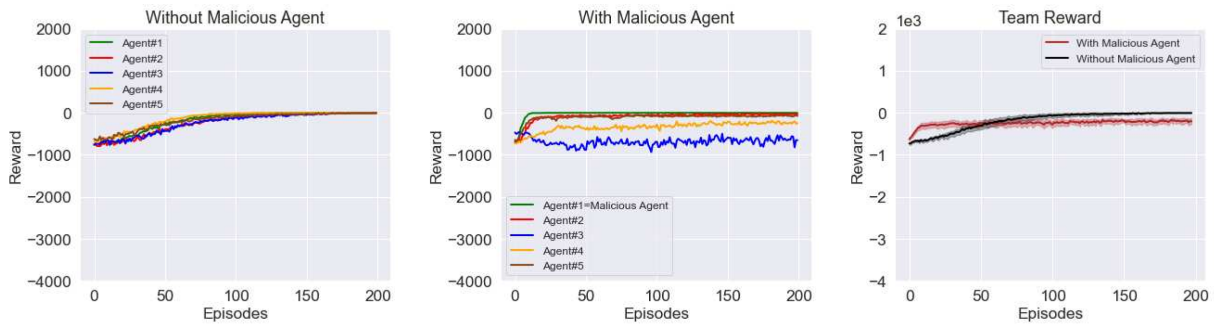
(c) Team reward with and without a malicious agent.

Figure 4.8: Reward convergence using the 5-norm immediate reward function during 200 episodes for $N = 5$ agents.



(a) Average reward without malicious agents. (b) Average reward with a malicious agent. (c) Team reward with and without a malicious agent.

Figure 4.9: Reward convergence using the Chebyshev immediate reward function during 200 episodes for $N = 5$ agents.



(a) Average reward without malicious agents. (b) Average reward with a malicious agent. (c) Team reward with and without a malicious agent.

Figure 4.10: Reward convergence using the combined immediate reward including Manhattan, Euclidean, 5-norm, and Chebyshev immediate reward functions during 200 episodes for $N = 5$ agents.

4.4.3 Modifying the Immediate Reward Function

The experiment is repeated with the same conditions but using the proposed Euclidean, n-norm ($n = 5$), Chebyshev, and combined immediate reward functions, (4.9)-(4.12).

As shown in Figure 4.7a and Figure 4.7b, the cumulative reward in both cases, without and with a malicious agent, have converged using the Euclidean immediate reward function in (4.9). The outcomes of using (4.9) is superior to the results of (4.8) because, as illustrated in Figure 4.7c, the MARL system's cumulative team reward with a malicious agent is larger than demonstrated results in Figure 4.6c. The cumulative team reward with a malicious agent has converged to -1468.74 using (4.9). Hence, as shown in Figure 4.6 and Figure 4.7, the use of (4.9) yields better results compared to (4.8).

We repeated the experiment using (4.10) where $n = 5$ (5-norm immediate reward function). As demonstrated in Figure 4.8, compared to the Figure 4.6 and Figure 4.7, the average received reward enhances for each agent and system by increasing n in the n-norm immediate reward function. For instance, the cumulative team reward with a malicious agent has converged to -1045.88 using (4.10), where $n = 5$.

As shown in Figure 4.9a and Figure 4.9b, the cumulative reward, without and with a malicious agent, have converged by applying the Chebyshev immediate reward function in (4.11). The results of using (4.11) are superior to (4.8)-(4.10), because, as illustrated in Figure 4.9c, the MARL system's cumulative team reward with a malicious agent is larger than demonstrated results in Figures 4.6c-4.8c. The cumulative team reward has converged to -369.11 using (4.11). Consequently, as displayed in Figures 4.6-4.9, the use of (4.11) has yielded more reliable results compared to (4.8)-(4.10). Using (4.11), the average received reward is higher for each agent and MARL system.

The outcomes of using (4.12) are superior to the results of (4.8)-(4.11), because, as shown in Figure 4.10c, the MARL system's cumulative team reward in the presence of a malicious agent is larger than illustrated results in Figures 4.6c-4.9c. The cumulative team reward has converged to -244.78 using (4.12). Hence, as demonstrated in Figure 4.10, the use of (4.12) has produced superior results compared to previously introduced immediate reward functions. Moreover, the average received reward is higher for each agent and system. The comparison of each agent's average cumulative reward as well as the average cumulative team reward using different immediate reward functions during 200 episodes without and with a malicious agent are indicated in Tables 4.1 and 4.2, respectively. As highlighted in these tables, the combined reward performed superior than the other rewards for each agent and team of agents.

Table 4.1: Comparison of each agent's average cumulative reward as well as the average cumulative team reward using different immediate reward functions during 200 episodes without a malicious agent.

	Agent 1	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team**
Manhattan Reward	−1566.21	−2011.39	−2093.50	−1209.28	−1490.39	−1641.68
Euclidean Reward	−756.45	−939.41	−961.17	−640.07	−992.70	−849.07
5-norm Reward	−576.56	−703.18	−720.72	−491.55	−655.37	−622.58
Chebyshev Reward	−330.13	−423.84	−430.31	−250.89	−300.96	−335.76
Combined Reward	−176.21	−231.49	−238.25	−135.51	−174.27	−185.91

Table 4.2: Comparison of each agent's average cumulative reward as well as the average cumulative team reward using different immediate reward functions during 200 episodes in the presence of a malicious agent.

	Agent 1	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team**
Manhattan Reward	−273.01	−2781.51	−3454.21	−2638.68	−2403.99	−2291.05
Euclidean Reward	−164.16	−1606.92	−2047.06	−1349.11	−2190.22	−1468.74
5-norm Reward	−143.52	−1351.53	−1258.49	−1179.52	−1305.38	−1045.88
Chebyshev Reward	−44.15	−143.68	−1062.88	−465.00	−173.53	−369.11
Combined Reward	−21.84	−87.28	−686.82	−338.84	−109.43	−244.78

4.4.4 The Immediate Rewards' Comparison After Normalization

To have a valid comparison between the used and proposed immediate reward functions, we normalize the accumulated reward values into a range of $[-1,0]$ for each agent using

$$R^i_{\mathrm{n}} = \frac{R^i - {R^i}_{\mathrm{min}}}{R^i_{\mathrm{max}} - {R^i}_{\mathrm{min}}} - 1, \tag{4.35}$$

where R^i and R^i_{n} are the cumulative reward and normalized cumulative reward vectors for i^{th} agent, respectively. Therefore, at this stage, the analysis is performed based on normalized data. Regarding Tables 4.3 and 4.4, as well as, Figures 4.11-4.15 after normalization, still the combined reward performed superior to the other rewards for each agent and team of agents (with and without malicious agents). It is worth mentioning that the data of Tables 4.3 and 4.4 are rounded to four decimal places. Furthermore, Figure 4.16 depicts the values of Tables 4.3 and 4.4 in two different charts. The performance of the malicious agent (`Agent#1`) in increasing its cumulative reward and decreasing the cumulative reward of the other agents is well illustrated in Figure 4.16b. In addition, Figure 4.16b shows how changing the type of immediate reward function can reduce the negative effect of the malicious agent.

Table 4.3: Comparison of each agent's average cumulative reward as well as the average cumulative team reward after normalization using different immediate reward functions during 200 episodes without a malicious agent.

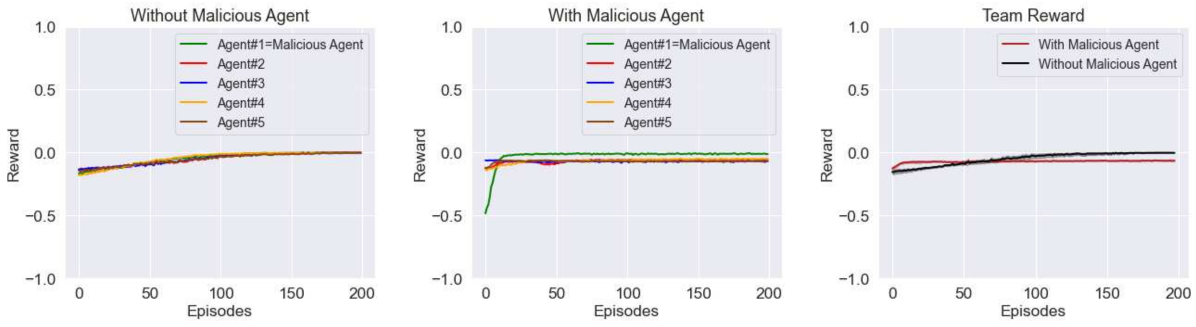
	Agent 1	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team**
Manhattan Reward	−0.0496	−0.0524	−0.0523	−0.0486	−0.0546	−0.0515
Euclidean Reward	−0.0489	−0.0522	−0.0521	−0.0475	−0.0545	−0.0510
5-norm Reward	−0.0449	−0.0498	−0.0500	−0.0431	−0.0518	−0.0479
Chebyshev Reward	−0.0396	−0.0452	−0.0455	−0.0371	−0.0447	−0.0424
Combined Reward	−0.0381	−0.0427	−0.0431	−0.0364	−0.0444	−0.0409

Table 4.4: Comparison of each agent's average cumulative reward as well as the average cumulative team reward after normalization using different immediate reward functions during 200 episodes in the presence of a malicious agent.

	Agent 1	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team**
Manhattan Reward	−0.0372	−0.0702	−0.0701	−0.0698	−0.0704	−0.0635
Euclidean Reward	−0.0334	−0.0688	−0.0702	−0.0686	−0.0702	−0.0623
5-norm Reward	−0.0249	−0.0659	−0.0702	−0.0645	−0.0697	−0.0590
Chebyshev Reward	−0.0153	−0.0366	−0.0686	−0.0561	−0.0406	−0.0435
Combined Reward	−0.0152	−0.0348	−0.0681	−0.0513	−0.0387	−0.0417

4.4.5 Reward Algorithm's Complexity and Execution Time

The comparison of the algorithm execution time of using different immediate reward functions during 200 episodes is presented in Table 4.5. Lower algorithm execution time and higher cumulative team reward are crucial factors in determining the type of the immediate reward function for the MARL system. By comparing the results of using Mr^i_{t+1} provided by [132], and the outcomes of using the proposed immediate reward functions (Er^i_{t+1}, Nr^i_{t+1}, $\check{C}r^i_{t+1}$, and Cr^i_{t+1}) the following results are obtained. The $\%\text{increase} = 100 \times \frac{\text{(final team reward - initial team reward)}}{|\text{initial team reward}|}$ is used to calculate the percentage increase of team reward, where initial team reward is the Manhattan team reward.

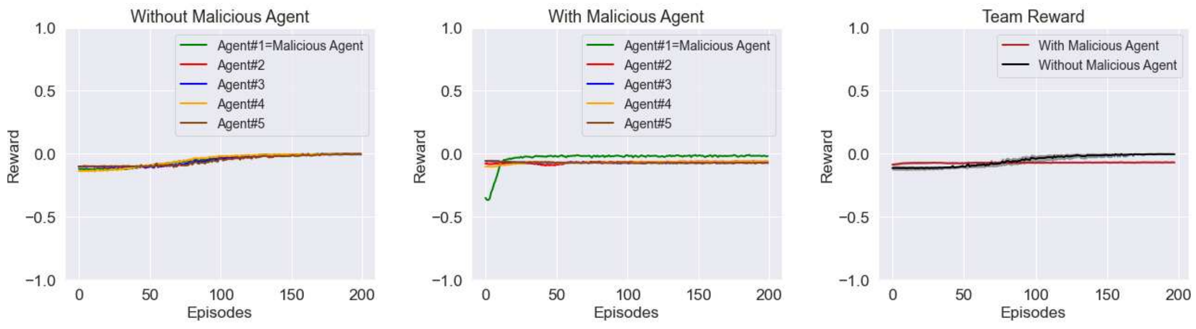


(a) Without malicious agents.

(b) With a malicious agent.

(c) Team reward with and without a malicious agent.

Figure 4.11: Normalized reward convergence using the Manhattan immediate reward function during 200 episodes for $N = 5$ agents.

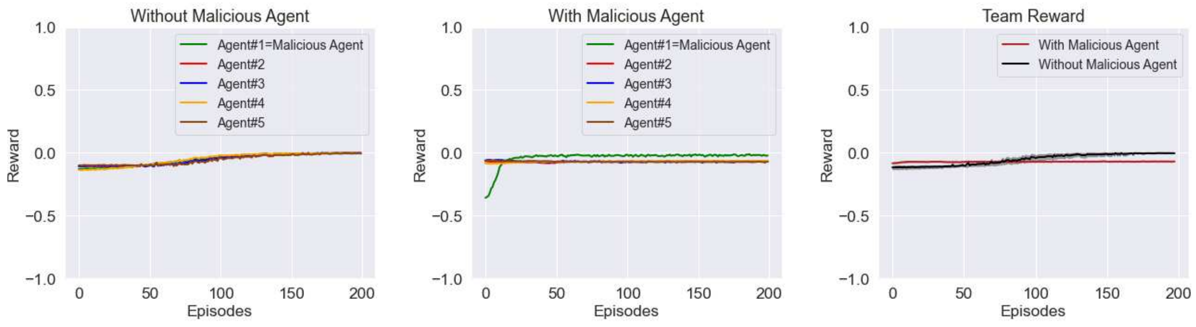


(a) Without malicious agents.

(b) With a malicious agent.

(c) Team reward with and without a malicious agent.

Figure 4.12: Normalized reward convergence using the Euclidean immediate reward function during 200 episodes for $N = 5$ agents.



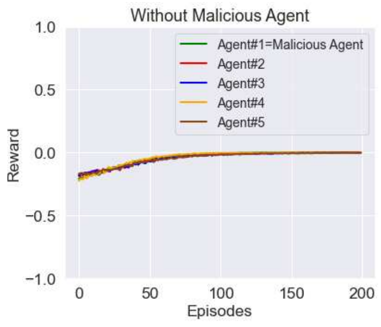
(a) Without malicious agents.

(b) With a malicious agent.

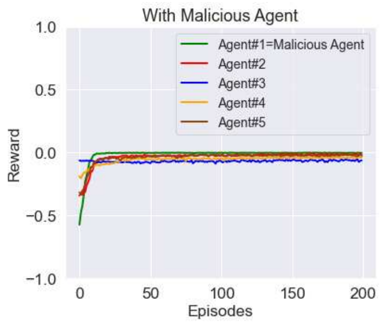
(c) Team reward with and without a malicious agent.

Figure 4.13: Normalized reward convergence using the 5-norm immediate reward function during 200 episodes for $N = 5$ agents.

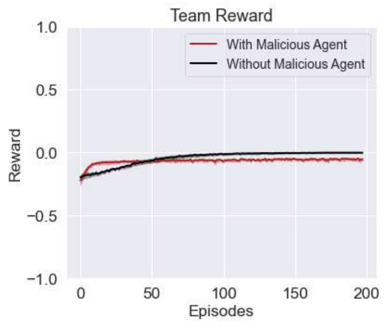
In addition, the Euclidean, 5-norm, Chebyshev, and combined team rewards are considered as the final team reward, each time.



(a) Without malicious agents.

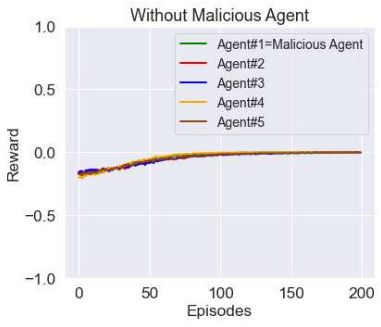


(b) With a malicious agent.

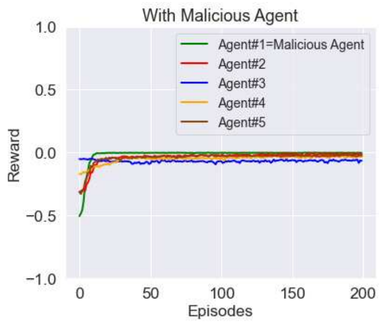


(c) Team reward with and without a malicious agent.

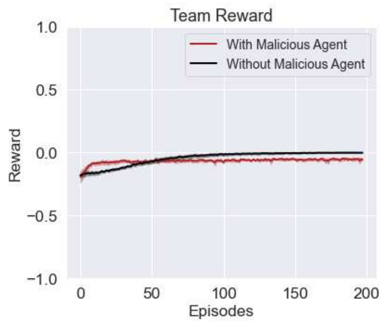
Figure 4.14: Normalized reward convergence using the Chebyshev immediate reward function during 200 episodes for $N = 5$ agents.



(a) Without malicious agents.

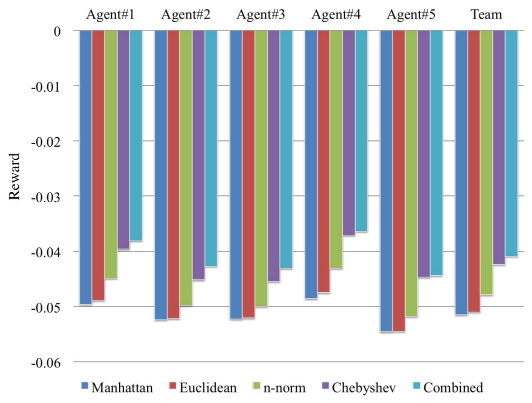


(b) With a malicious agent.

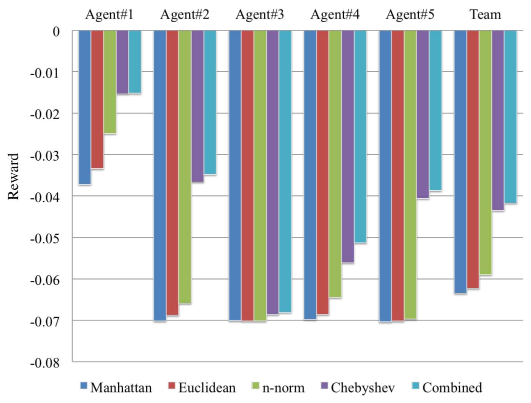


(c) Team reward with and without a malicious agent.

Figure 4.15: Normalized reward convergence using the combined immediate reward including Manhattan, Euclidean, 5-norm, and Chebyshev immediate reward functions during 200 episodes for $N = 5$ agents.



(a) Normalized average cumulative reward without malicious agents.



(b) Normalized average cumulative reward with a malicious agent (Agent#1).

Figure 4.16: Normalized average cumulative reward for each agent and a team of agents, including $N = 5$ agents, using various immediate reward functions during 200 episodes.

4.4.5.1 Before Normalization

By comparing the results of using Manhattan and Euclidean immediate rewards, it is concluded that after using the Euclidean immediate reward, the $+48.28\%$ and $+35.89\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the Euclidean immediate reward function is $+1.76$ times that of the Manhattan immediate reward function. Moreover, by comparing the outcomes of using Manhattan and 5-norm immediate rewards, it is realized that after using the 5-norm immediate reward, the $+62.08\%$ and $+54.35\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the 5-norm immediate reward function is $+1.28$ times that of the Manhattan immediate reward function. Furthermore, by comparing the outcomes of using Manhattan and Chebyshev immediate rewards, it is achieved that after using the Chebyshev immediate reward, the $+79.55\%$ and $+83.89\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the Chebyshev immediate reward function is $+1.08$ times that of the Manhattan immediate reward function. Besides, by comparing the results of using Manhattan and combined immediate rewards, it is concluded that after using the combined immediate reward, the $+88.68\%$ and $+89.32\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the combined immediate reward function is $+3.14$ times that of the Manhattan immediate reward function.

4.4.5.2 After Normalization

By comparing the results of using Manhattan and Euclidean immediate rewards, it is concluded that after using the Euclidean immediate reward, the $+0.97\%$ and $+1.89\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the Euclidean immediate reward function is $+1.75$ times that of the Manhattan immediate reward function. Moreover, by comparing the outcomes of using Manhattan and 5-norm immediate rewards, it is realized that after using the 5-norm immediate reward, the $+6.99\%$ and

$+7.09\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the 5-norm immediate reward function is $+1.29$ times that of the Manhattan immediate reward function. Furthermore, by comparing the outcomes of using Manhattan and Chebyshev immediate rewards, it is noted that after using the Chebyshev immediate reward, the $+17.67\%$ and $+31.50\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the Chebyshev immediate reward function is $+1.08$ times that of the Manhattan immediate reward function. Besides, by comparing the results of using Manhattan and combined immediate rewards, it is concluded that after using the combined immediate reward, the $+20.58\%$ and $+34.33\%$, increase in the assertiveness of team reward without and with a malicious agent, respectively. The algorithm execution time using the combined immediate reward function is $+2.95$ times that of the Manhattan immediate reward function.

Table 4.5 lists the time complexity of various types of immediate reward algorithms. The time complexity of Manhattan, Euclidean, 5-norm, and Chebyshev immediate reward functions with n point pairs are $O(n)$, and they take linear time. Moreover, the time complexity of the combined immediate reward algorithm is $O(n)$ as well.

Nevertheless, this time difference would not mean that the proposed immediate rewards are better or worse. Still, with longer episodes having more time-steps, this execution time difference may be significant. Note that the data of Table 4.5 are rounded to two decimal places.

It is worth mentioning that for developing and simulating this chapter's algorithm, Python and MATLAB Programming Languages are utilized.

4.5 Conclusions

We studied the consensus problem of a leaderless, homogeneous MARL system using the actor-critic algorithms in the absence and presence of malicious agents. Each agent's principal goal is

Table 4.5: Comparing the results of algorithm's complexity and execution time using different immediate reward functions.

	Manhattan Reward	**Euclidean Reward**	**5-norm Reward**	**Chebyshev Reward**	**Combined Reward**
Algorithm Execution Time(seconds)	31.41 ± 0.02	55.27 ± 0.03	40.07 ± 0.42	33.98 ± 0.20	98.67 ± 0.45
Algorithm Execution Time(seconds) (Normalized)	31.23 ± 0.13	54.57 ± 0.04	40.18 ± 0.01	33.72 ± 0.09	92.22 ± 0.19
Algorithm Time Complexity	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

to reach the position consensus with the maximum cumulative reward. We presented the immediate reward function based on Manhattan distance. Then, we proposed three other immediate reward functions based on various distance metrics to improve the MARL system's performance. We combined various immediate reward functions and used each of them based on the maximum returned value during each episode to enhance agents' cumulative reward in the presence of malicious agents within the MARL system. Finally, we compared different immediate reward functions within the MARL system and we found that the type of immediate reward function plays a significant role in efficiency of each agent in the network in reaching the consensus and obtaining further cumulative team reward.

Chapter 5

Control of Multi-agent Reinforcement Learning Systems: The Effect of Neural Network Structure

5.1 Introduction

This chapter is a continuation of Chapter 4 with the same research background to reduce the malicious agent's adverse effects on a MARL system, including actor-critic architecture. In this chapter, an attempt has been made to achieve the overall goal of the MARL system, which is to increase the cumulative reward of all individual agents and reduce the malicious agents' negative effect on the entire MARL system. For this purpose, considering that the adverse agent is detectable, we have changed the malicious agent's NN structure. The claims have been proven theoretically, and the simulation confirms theoretical findings. The methodology we have used to prove the superiority of a NN structure over another NN architecture in terms of the amount of loss is the gradient of the loss function with respect to the activation function.

5.2 Background

The temporal difference (TD) error defines the comparison between the predicted reward $\hat{r}_{t+1}$ and the actual reward r_{t+1}. Higher TD error values are associated with greater differences between actual reward r_{t+1} and predicted reward $\hat{r}_{t+1}$. The discussed TD error δ_t, is given as follows

$$\delta_t = R_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), \tag{5.1}$$

where R_{t+1} is the reward function which is given by

$$R_{t+1} = \mathbb{E}\left[r_{t+1} | s_t = s, a_t = a\right]. \tag{5.2}$$

Moreover, $V_t(s_t)$ is the critic value function at time t that is specified as below

$$V_t(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_t = s\right]. \tag{5.3}$$

The critic value function $V_{t+1}(s_t)$ at time $t+1$ is

$$V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t, \tag{5.4}$$

using the learning rate α. The consensus of N agents is given by

$$\begin{cases} \lambda_{t+1}^i \leftarrow \sum_{j \in \mathcal{N}} w_t(i,j) \cdot \tilde{\lambda}_t^j, \\ v_{t+1}^i \leftarrow \sum_{j \in \mathcal{N}} w_t(i,j) \cdot \tilde{v}_t^j, \end{cases} \tag{5.5}$$

where λ and v are the actual multi-agent reward function parameter, and actual multi-agent value function parameter, respectively. Moreover, $\tilde{\lambda}$ and $\tilde{v}$ are the predicted multi-agent reward function parameter and the predicted multi-agent value function parameter, respectively. The initialization of λ, $\tilde{\lambda}$, v, and $\tilde{v}$ parameters is done for all N agents at time $t = 0$ [128], [132]. At time $t+1$, the above parameters should be updated and added to the previous value list. The network value function approximation $V_t^i(s_t;v_t^i)$ is characterized by v^i of each agent. In this regard, the following is achieved

$$\tilde{\lambda}_t^i \leftarrow \lambda_t^i + \alpha_{v,t}\left(r_{t+1}^i - \hat{r}_{t+1}\left(\lambda_t^i\right)\right)\nabla_\lambda \hat{r}_{t+1}\left(\lambda_t^i\right) \tag{5.6}$$

$$\tilde{v}_t^i \leftarrow v_t^i + \alpha_{v,t}\delta_t^i \nabla_v V_t^i\left(s_t;v_t^i\right). \tag{5.7}$$

The set of neighbors of the i^{th} agent is described by $\mathscr{N}$, and the Metropolis weight matrix specified by $W_t = [w_t(i,j)]_{N\times N}$ is

$$W_t = \begin{cases} \dfrac{1}{1+\max\{d_t(i),d_t(j)\}} & \text{if } (i,j)\in\mathscr{E}, \\ 1-\displaystyle\sum_{(i,k)\in\mathscr{E}} W_t(i,k) & \text{if } i=j, \\ 0 & \text{otherwise,} \end{cases} \tag{5.8}$$

where the degrees of agents i and j are defined by $d_t(i)$ and $d_t(j)$, respectively. Furthermore, at time t, $w_t(i,j)$ indicates the weight on the message transferred from agent i to agent j. MARL's position consensus requires all N agents to perform the consensus step (5.5). To update the i^{th} agent's policy function $\pi_i(a_t^i|s_t)$ the reward function parameter λ^i and value function parameter v^i should be updated.

A loss function describes the model's performance according to the current set of parameters (weights and biases). The MSE loss function is given by

$$\text{MSE} = \frac{1}{n}\sum_{k=1}^{n}\left(\hat{y}_k - y_k\right)^2, \tag{5.9}$$

where y_k is the actual value and $\hat{y}_k$ is the estimated value, using k^{th} sample out of n samples. Any machine learning (ML) model aims to minimize the loss function. The gradient descent method is utilized, as an iterative optimization method, to minimize the loss function in ML and determine the most appropriate parameters.

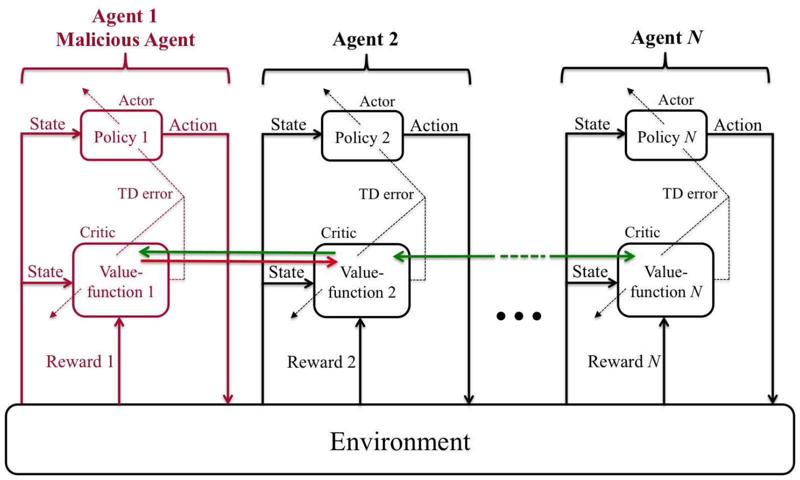


Figure 5.1: An illustration of a multi-agent actor-critic architecture including a malicious agent (red agent). The correct data between the neighboring agents are transferred via green arrows, and inaccurate data from the adverse agent is transmitted to neighboring agents through the red arrow.

5.3 Methodology

To mitigate the harmful effects of malicious agents, we modify the adverse agents' NN architectures, especially the activation function of the last layer, to enhance the cumulative loss and decrease the cumulative reward of the malicious agents. A multi-agent actor-critic system with a malicious agent is demonstrated in Figure 5.1.

5.3.1 Modifying the Neural Network Structure

To train the i^{th} agent's critic and reward networks in [132], and [206], the authors have used a fully connected NN structure containing m dense layers. Each layer of this NN includes n neurons. The last layer of each NN has a linear activation function in combination with the MSE loss function. According to the properties of a linear function, it is evident that the output of this function is in the range of $(-\infty, +\infty)$, which makes the uncertain activation bound range. Moreover, the derivative of a linear function is a constant value. Consequently, the gradient with respect to the variable is constant. Therefore, descent converges to a constant gradient in gradient descent for training. When there is an error between the actual and predicted values, the back-propagation (feedback) changes are constant and do not depend on the calculated error. Therefore, we replace the sigmoid activation function with the linear function in the last layer of malicious agents' critic and reward NN. Afterward, we analyze the results. The main reason for this choice is that sigmoid is a nonlinear function, and any combination with that becomes nonlinear. Thus, the sigmoid function's gradient is smooth and non-constant. In addition, the output of this function is in the range of $(0,+1)$ or $(-1,+1)$, which makes the specific activation bound range. The specificity of the output boundary indicates that the activation bound is in a particular range that prevents the activation from exploding.

Assumption 5.1 *The malicious agents have already been detected.*

Theorem 5.1 *Given Assumption 5.1, the combination of the MSE loss function with the sigmoid activation function provides a higher gradient of loss than its combination with the linear activation function.*

Proof 5.1 *The last layer's loss function of malicious agent's critic and reward NN, including n neurons, is the MSE loss function that is given by*

$$\ell = \frac{1}{n}\sum_{k=1}^{n}(\hat{r}_k - r_k)^2, \tag{5.10}$$

where r and $\hat{r}$ are actual and predicted rewards, respectively. Furthermore, the linear activation function is considered as

$$g(x) = x. \tag{5.11}$$

Consequently, the combination of MSE loss function (5.10) and linear activation function (5.11) in the regression setting is given by

$$\ell(\hat{r}, g(x)) = \frac{1}{n}\sum_{k=1}^{n}(\hat{r}_k - g(x))^2. \tag{5.12}$$

In this regard, the gradient of MSE loss function with respect to linear activation function is a *linear function as follows*

$$\begin{aligned}
\frac{\partial \ell(\hat{r}, g(x))}{\partial x} &= \frac{\partial \ell(\hat{r}, g(x))}{\partial g(x)} \cdot \frac{\partial g(x)}{\partial x} \\
&= \frac{\partial}{\partial g(x)} \left(\frac{1}{n} \sum_{k=1}^{n} (\hat{r}_k - g(x))^2 \right) \cdot \frac{\partial}{\partial x} (x) \\
&= \frac{2}{n} \sum_{k=1}^{n} (g(x) - \hat{r}_k) \\
&= \frac{2}{n} \sum_{k=1}^{n} (x - \hat{r}_k) .
\end{aligned} \tag{5.13}$$

The MSE loss function with the combination of sigmoid activation function is utilized at the last layer of the malicious agent's critic and reward NN. The sigmoid activation function is given by

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{5.14}$$

The combination of MSE loss function and sigmoid activation function in the regression setting is given by

$$\ell(\hat{r}, \sigma(x)) = \frac{1}{n} \sum_{k=1}^{n} (\hat{r}_k - \sigma(x))^2 . \tag{5.15}$$

Therefore, the gradient of MSE loss function with respect to sigmoid activation function is a non-

linear function as below

$$\begin{aligned}
\frac{\partial \ell(\hat{r},\sigma(x))}{\partial x} &= \frac{\partial \ell(\hat{r},\sigma(x))}{\partial \sigma(x)} \cdot \frac{\partial \sigma(x)}{\partial x} \\
&= \frac{\partial}{\partial \sigma(x)} \left(\frac{1}{n} \sum_{k=1}^{n} (\hat{r}_k - \sigma(x))^2 \right) \cdot \frac{\partial}{\partial x} \left(\frac{1}{1+e^{-x}} \right) \\
&= \frac{2}{n} \sum_{k=1}^{n} (\hat{r}_k - \sigma(x)) \cdot \frac{-e^{-x}}{(1+e^{-x})^2} \cdot \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{2}{n} \sum_{k=1}^{n} (\sigma(x) - \hat{r}_k) \cdot \left(\frac{1}{1+e^{-x}} \left(1 - \frac{1}{1+e^{-x}} \right) \right)^2 \\
&= \frac{2}{n} \sum_{k=1}^{n} (\sigma(x) - \hat{r}_k) \cdot \left(\frac{1}{1+e^{-x}} \right)^2 \cdot \left(1 - \frac{1}{1+e^{-x}} \right)^2 \\
&= \frac{2}{n} \sum_{k=1}^{n} (\sigma(x) - \hat{r}_k) \cdot \sigma^2(x) \cdot (1 - \sigma(x))^2.
\end{aligned} \tag{5.16}$$

For the malicious agent, the episode reward $r_{episode}$ is considered as a negative episode reward by default; therefore, the obtained result of Equation (5.16) yields a higher gradient of loss rather than the obtained result of Equation (5.13) as below

$$\frac{2}{n} \sum_{k=1}^{n} (g(x) - \hat{r}_k) < \frac{2}{n} \sum_{k=1}^{n} (\sigma(x) - \hat{r}_k) \cdot \sigma^2(x) \cdot (1 - \sigma(x))^2. \tag{5.17}$$

$$\sum_{k=1}^{n} (g(x) - \hat{r}_k) < \sum_{k=1}^{n} (\sigma(x) - \hat{r}_k) \cdot \sigma^2(x) \cdot (1 - \sigma(x))^2. \tag{5.18}$$

Hence, the Theorem 5.1 is proven.

Considering the Theorem 5.1, and from Inequality (5.18), it is expected that the combination of MSE loss function with linear activation function yields a lower cumulative loss than the combination of MSE loss function with a sigmoid activation function. As a hint, lower cumulative loss provides higher cumulative reward. The NN is trained using the gradient of the MSE loss function two times– one time utilizing the gradient of the MSE loss function with respect to the linear activation function $g(x)$, and another time using the gradient of MSE loss function concerning the sigmoid activation function $\sigma(x)$. After specific episodes, the cumulative loss combining both linear and sigmoid activation functions started to decrease. However, the speed of loss reduction in combination with the linear activation function is higher than the speed of loss reduction in combination with the sigmoid activation function. In the meantime, the cumulative loss using both activation functions is calculated. Hence, the cumulative loss with the sigmoid activation function provides a higher value than the cumulative loss with the linear activation function.

It is worth mentioning that Subsection 5.3.1 is done to enhance the cumulative loss and reduce the cumulative reward of malicious agents. In this regard, the adverse effect of the malicious agent is decreased by increasing the loss.

5.4 Results and Discussion

For obtaining the results, the combined immediate reward function r^i_{t+1} (one of the proposed immediate reward functions of Chapter 4) is given by

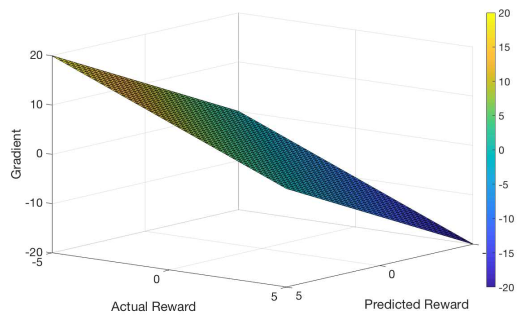
$$r^i_{t+1} = \max\left(Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, \check{C}r^i_{t+1}\right), \tag{5.19}$$

where Mr^i_{t+1}, Er^i_{t+1}, Nr^i_{t+1}, and $\check{C}r^i_{t+1}$ are the Manhattan, Euclidean, n-norm, and Chebyshev immediate reward functions, respectively [3]. Furthermore, in training, the distance metric that accompanies the combined immediate reward function is Euclidean distance

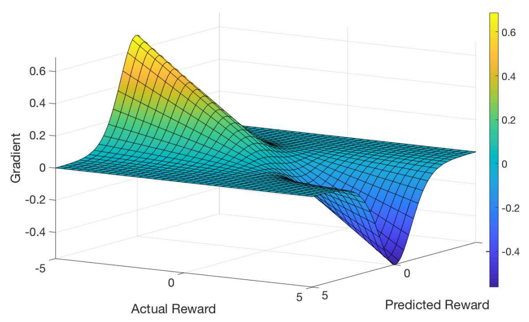
$$E_d = \left(\left|x^i - x^i_{des}\right|^2 + \left|y^i - y^i_{des}\right|^2\right)^{1/2}, \tag{5.20}$$

where (x^i, y^i) and (x^i_{des}, y^i_{des}) are the current position and the desired position of the i^{th} agent, respectively.

We have used and extended a part of the code provided in [206] for a part of our implementation. Moreover, the algorithm's execution is done using a system with 3.60 GHz Intel Core i7 $-$ 7700 processor, 16 GB installed RAM, 64$-$bit operating system, and x64$-$based processor.



(a) The gradient of MSE loss function with respect to linear activation function.



(b) The gradient of MSE loss function with respect to sigmoid activation function.

Figure 5.2: The gradient of MSE loss function with respect to linear activation function and sigmoid activation function for $n = 30$ neurons in the range of $[-20, 20]$ and $[-0.6, 0.6]$, respectively.

5.4.1 Consequences of Modifying the Neural Network Structure

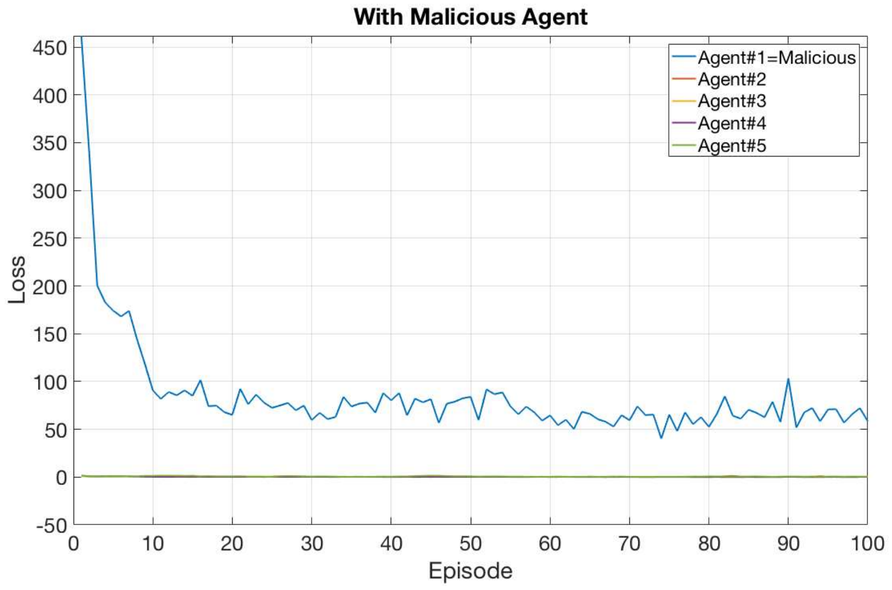
Figure 5.2 shows the gradient of the MSE loss function with respect to two different activation functions. The gradient of MSE loss function ℓ with respect to linear activation function $g(x)$ (final obtained Equation (5.13)) is illustrated in Figure 5.2a for $n = 30$ neurons. Moreover, Figure 5.2b demonstrates the gradient of MSE loss function ℓ with respect to sigmoid activation function $\sigma(x)$ (final generated Equation (5.16)) for $n = 30$ neurons.

5.4.1.1 MSE Loss Function with Linear Activation Function

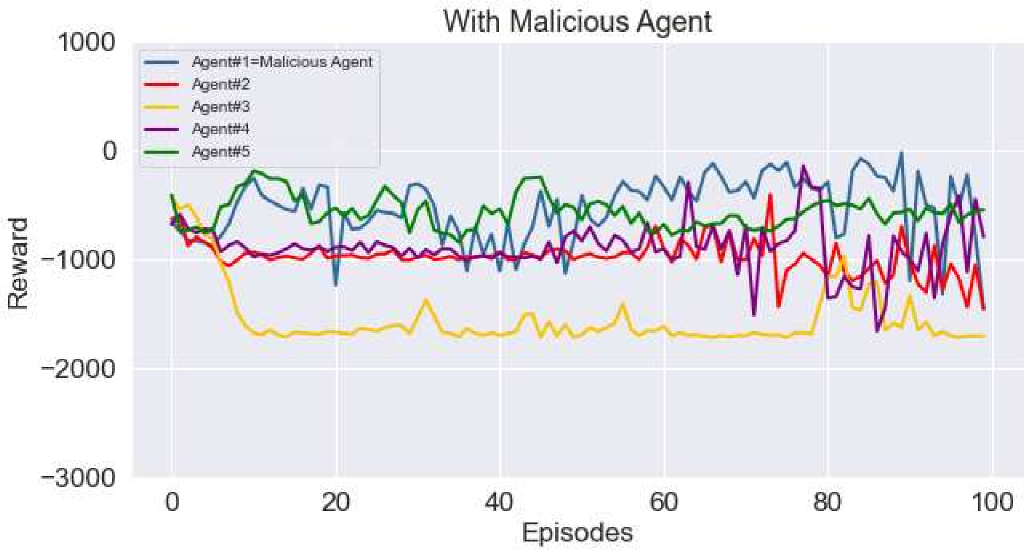
In the last layer of all agents' critic and reward NN (malicious and non-malicious agents), the combination of MSE loss function and linear activation function is used. The simulation results are demonstrated in Figure 5.3. As shown in Figure 5.3a, the average loss of non-malicious agents are close to zero. In addition, the average loss of malicious agent tends to zero over time, and thus the average reward of this adverse agent tends to enhance (Figure 5.3b). The tendency to increase the reward affects non-malicious agents' performance negatively.

5.4.1.2 MSE Loss function with Linear and Sigmoid Activation Function

In the last layer of malicious agent's critic and reward NN, the MSE loss function and sigmoid activation function are combined. However, for the rest of the agents (non-malicious agents), the



(a) Average loss convergence, using linear activation function at the last layer of all agents' NN.



(b) Average reward convergence, using linear activation function at the last layer of all agents' NN.

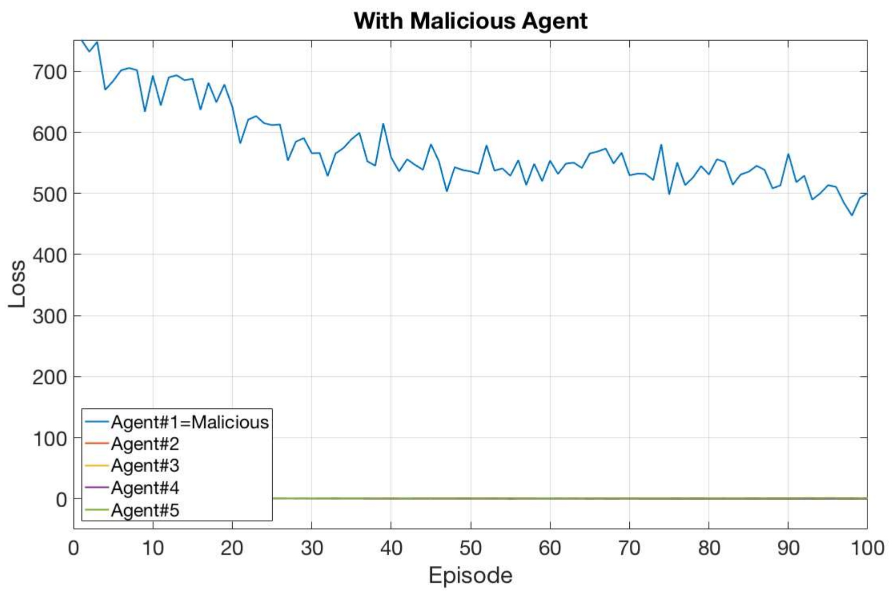
Figure 5.3: Average loss and reward convergence for $N = 5$ agents, including a malicious agent (Agent#1), during 100 episodes and 1000 time-steps.

integration of the MSE loss function and linear activation function is applied in the critic and reward NN. The simulation results are illustrated in Figure 5.4. As shown in Figure 5.4a, the average loss of non-malicious agents are close to zero. However, the average loss of malicious agent tends to a value in the range of $(500, 600)$ over time. Therefore, the average reward of the adverse agent tends to decrease (Figure 5.4b). To some extent, the tendency to decrease the reward affects non-malicious agents' performance positively.

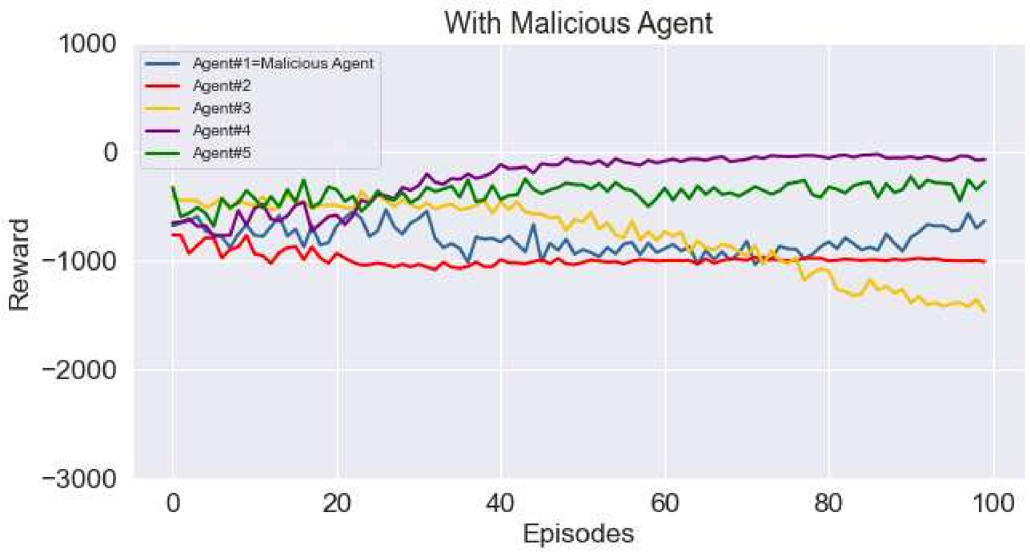
Table 5.1 shows that when the MSE loss function with sigmoid activation function is combined in the malicious agent's critic and reward NN (last layer), the average loss of `Agent#1` has been increased dramatically from 84.2749 to 574.3421 compared to the situation when the linear activation function is used in the last layer of all agents' critic and reward NN. Due to the limitation of the sigmoid activation bound range of malicious agent and growing the loss of `Agent#1`, it is expected that its average cumulative reward will decrease.

Table 5.2 represents that by combining the MSE loss function and sigmoid activation function (in the last layer of malicious agent's critic and reward NN), the adverse agent's average cumulative reward has been reduced from -528.11 to -806.12 compared to the case when the linear activation function is used in the last layer of all agents' critic and reward NN. Moreover, using the sigmoid activation function at the last layer of critic and reward NN, the team reward of non-malicious agents is increased from -992.26 to -591.89.

As can be seen from the charts shown in Figure 5.5, using the sigmoid activation function in the NN structure of the malicious agent (Agent#1) provides a greater loss than in the case of linear activation function (Figure 5.5a). In addition, according to Figure 5.5b, the reward for malicious agent when using the sigmoid activation function in its NN architecture is less than when using the linear activation function. Consequently, using the sigmoid activation function in the malicious agent NN structure increases the reward in other agents. Note that the linear activation function is still used in the NN architecture of other agents.



(a) Average loss convergence, using sigmoid activation function and linear activation function at the last layer of malicious agent and non-malicious agents' NN, respectively.



(b) Average reward convergence, using sigmoid activation function and linear activation function at the last layer of malicious agent and non-malicious agents' NN, respectively.

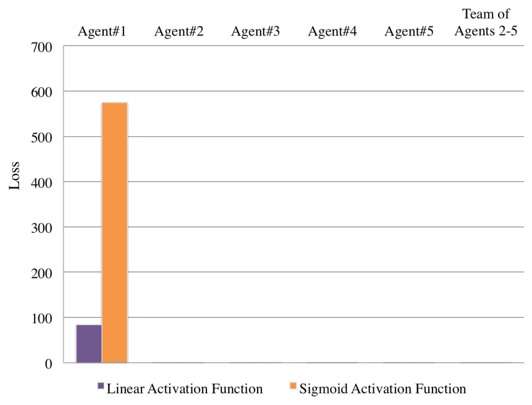
Figure 5.4: Average loss and reward convergence for $N = 5$ agents, including a malicious agent, during 100 episodes and 1000 time-steps.

Table 5.1: Comparison of each agent's average loss using linear and sigmoid activation functions at the last layer of malicious agent's (`Agent#1`) critic and reward NN during 100 episodes.

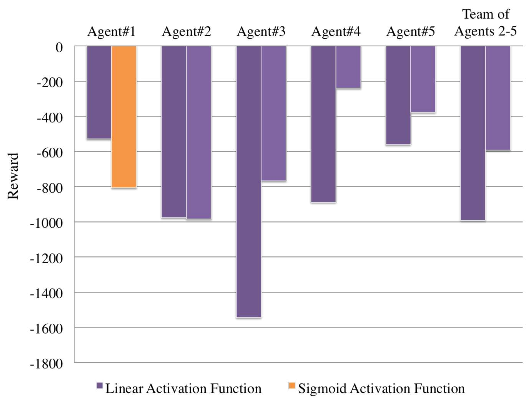
Last Layer Activation Function \ **Agents**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team (Agents 2-5)**
Linear: All Agents Critic and Reward NN	84.2749	0.3557	0.2961	0.2652	0.6500	0.3917
Sigmoid: Malicious Agent Linear: Other Agents Critic and Reward NN	574.3421	0.3698	0.5017	0.2687	0.9482	0.5221

Table 5.2: Comparison of each agent's average cumulative reward using linear and sigmoid activation functions at the last layer of malicious agent's (`Agent#1`) critic and reward NN during 100 episodes.

Last Layer Activation Function \ **Agents**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team (Agents 2-5)**
Linear: All Agents Critic and Reward NN	-528.11	-975.90	-1544.38	-888.20	-560.59	-992.26
Sigmoid: Malicious Agent Linear: Other Agents Critic and Reward NN	-806.12	-984.28	-766.80	-239.37	-377.13	-591.89



(a) Loss of individual agents and a team of agents (Agents#2-5).



(b) Reward of individual agents and a team of agents (Agents#2-5).

Figure 5.5: Loss and reward for each agent and a team of agents 2-5, including $N = 5$ agents, using linear and sigmoid activation functions during 100 episodes and 1000 time-steps.

It is concluded that manipulating and modifying the NN structure (using a nonlinear activation function instead of a linear activation function) of malicious agents makes it possible to decrease their adverse effects on the MARL performance.

Python and MATLAB Programming Languages are used to develop and simulate the results in this chapter.

5.5 Conclusions

In this chapter, we studied the control of a leaderless, homogeneous MARL system using actor-critic algorithms in the presence of a malicious agent. Using the gradient of the loss function with respect to the activation function, we proved that when the MSE loss function is combined with the sigmoid activation function in the malicious agent's critic and reward NN (last layer), the loss of `Agent#1` (malicious agent) is increased and the cumulative reward is decreased dramatically compared to the situation when the linear activation function is used in the last layer of adverse agent's critic and reward NN.

Chapter 6

Adversarial Attacks on Heterogeneous Multi-agent Deep Reinforcement Learning System with Time-delayed Data Transmission

6.1 Introduction

This chapter studies the gradient-based adversarial attacks on cluster-based, heterogeneous, MADRL systems with time-delayed data transmission. The structure of the MADRL system consists of various clusters of agents. The DQN architecture presents the first cluster's agent structure. The other clusters are considered as the environment of the first cluster's DQN agent. We introduce two novel observations in data transmission, termed on-time and time-delay observations. The proposed observations are considered when the data transmission channel is idle and the data is transmitted on-time or time-delayed. Considering the distance between the neighbouring agents, we present a novel immediate reward function by appending a distance-based reward to the previously utilized reward to improve the MADRL system performance. We consider three types of gradient-based attacks to investigate the robustness of the proposed system data transmission. Two defence methods are proposed to reduce the effects of the discussed malicious attacks. We have rigorously shown the system performance based on the DQN loss and the team reward for the entire team of agents. Moreover, the effects of the various attacks before and after using defence algorithms are demonstrated. The theoretical results are illustrated and verified with simulation examples. The proposed algorithms and methods in this chapter can be used for data transmission between agents of a MARL system or a MADRL system to reach a consensus related to the Chapters 4 and 5.

6.2 Background

Decision-making is based on the information received from the environment by an RL or DRL agent. It is considered that the finite MDP represents the dynamics of the environment for decision-making. The 5-tuple $M = \langle s, a, T, R, \gamma \rangle$ presents an MDP for an RL and DRL system, where s is a finite set of environmental states, and a is a finite action set. Moreover, $T(s_t, a_t, s_{t+1}) \rightarrow [0,1]$ is the state-transition probability function that agent takes action a_t in the state s_t, and is transferred to the state s_{t+1} to do the next action. Further, $R(s_t, a_t, s_{t+1}) \triangleq \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \rightarrow \mathbb{R}^n$ is a cumulative reward function, where r_{t+k+1} shows the immediate reward, with discount factor γ that is the trade-off between an immediate reward and potential future reward.

In the leaderless MAS scenario, all agents communicate with their cluster-mates, as well as agents of other clusters. In the leader-follower MAS scenario, in each cluster, only the preassigned leader communicates with the other agents in the same cluster as well as leaders of different clusters. Thus, data transmission occurs between the leader and the followers of one cluster as well as leaders of clusters. The leaderless and leader-follower MAS are considered as the graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of all agents, and $\mathscr{E} \subseteq \{(i,j) | i \in \mathscr{V}, j \in \mathscr{V}\}$ is the set of all communication links between agents. The agents i and j communicate if and only if $(i,j) \in \mathscr{E}$ [207].

6.3 Methodology

In this section, the leaderless and leader-follower topologies are introduced. Then, the components of the DQN algorithm (observation, action, state, and reward) are explained. Afterward, the DQN loss for on-time and time-delayed data transmission is justified. Three types of adversarial attacks to target the proposed leader-follower MAS (state of the DQN agent) are extended and explained. Finally, two defence methods against performed adversarial attacks are introduced.

6.3.1 Leaderless and Leader-follower Topologies

A generic illustration of the MADRL system topology including *N* static, heterogeneous agents, and *P* clusters is shown in Figure 6.1. The leaderless and leader-follower MAS scenarios can be envisioned from the presented topology in Figure 6.1. The goal of each static agent in this topology is to transfer data with the maximum average reward.

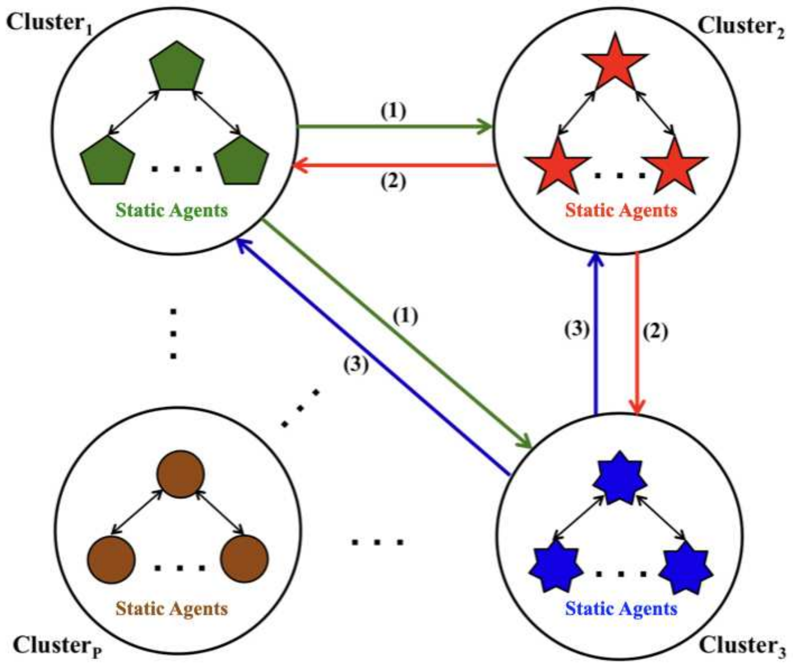


Figure 6.1: An illustration of the MADRL system topology, including *N* static, heterogeneous agents and *P* clusters.

6.3.2 Observation

In this chapter, observation describes the state of the data transmission channels that are either idle or busy [163]. If the channel between each pair of agents is busy at time step t, no data can be transmitted at time step $t+1$ due to data transition by another agent. However, if this channel is idle, data can be transmitted. The transmitted data either reaches its destination successfully or collides along the way, gets corrupted, and does not reach the goal. Therefore, the defined observation set by [163] is $o_t = \{\text{busy, idle, successful, collided}\}$.

We propose a modified observation set to use in our MADRL system. We add on-time and time-delayed arrival states to the observation set. As in [163], it is first checked that the data channel between each pair of agents is either idle or busy. If the channel is idle, the data can be transferred successfully on-time, successfully with time-delay, or collided. Therefore, in the introduced scenarios of this chapter, the novel observation set $o_t = \{\text{busy, idle, on-time, time-delay, collided}\}$ is proposed.

The lengths of the transferred packets in the network are different and belong to the set of $R_c \in \{1,2,\ldots,R_{c\max}\}$. When the observation is on-time, it means that each agent in the network transmits the packet at the next R_c mini-slot, with the action time duration in the set of $T_d(a_t) \in \{1,2,\ldots,R_{c\max}\}$. With the time-delay observation, each agent in the network transmits the packet at the next R_c mini-slot, with the action time duration in the set of $T'_d(a_t) \in \{R_{c\max}+1, R_{c\max}+2,\ldots\}$. In both on-time and time-delay observations, when an agent transmits a packet in a data transmission channel, no other agent sends the data at that specified channel to avoid a collision. In the following, $T_d(a_t)$ and $T'_d(a_t)$ are abbreviated as T_d and T'_d, respectively. When the observation is collided, it means that the agent transmits the packet at the next R_c mini-slot; however, another agent transmits data in at least one of the R_c mini-slots. Note that each mini-slot is a required time to perform CSMA. In this chapter, each mini-slot is considered a time step.

6.3.3 Action

Action is one of the significant components of RL and DRL algorithms [208]. In general, the agent receives the corresponding state from the environment at time step t and performs the appropriate action accordingly. Due to the quality of the performed action at time step t, the agent receives the reward associated with that action at time step $t+1$. According to the observation set in this chapter, the agent first checks whether the data channel is idle or busy. This stage should be done at less than one mini-slot. The performed actions in this chapter are based on [163], as follows:

- *No Selection:* If the channel is busy at time step t (checked at less than one mini-slot), the DQN agent does not take any action at the next time step. Hence, $a_{t+1} = 0$.

- *Uniform Selection with Probability ε:* If the channel is idle at time step t, the DQN agent chooses an action (transfer or not to transfer a packet) at time step $t+1$. If the agent at the next R_c mini-slot transmits packets with the length of R_c, then the action at time step $t+1$ is $a_{t+1} = R_c$, where $R_c \in \{0, 1, 2, \ldots, R_{c\max}\}$. This action selection method is a uniform random election with probability ε (exploration) using ε-greedy algorithm.

- *Non-uniform Selection with Probability $1-\varepsilon$:* If the channel is idle at time step t, an action to transfer or not to transfer a packet at time step $t+1$ can be chosen by the DQN agent. According to the conventional ε-greedy DQN algorithm, the action will be the maximum Q-value $\{Q(s, a; \theta) | a \in \mathbb{A}\}$, where Q is a parametric function including state s, action a, and parameter θ as a vector, including the weights in the NN. Moreover, $\mathbb{A}$ is the set of actions. Therefore, the action at time step $t+1$ is

$$a_{t+1} = \underset{a_t \in \{0,1,2,\ldots,R_{c\max}\}}{\arg\max} \sum_{i=1}^{N} Q^i \left(s_{t+1}, a_t; \theta^-\right), \tag{6.1}$$

  where θ^- denotes the target Q-value weight. This action selection method is a non-uniform selection with probability $1-\varepsilon$ (exploitation) using ε-greedy algorithm.

Note that the ε-greedy algorithm is a widely used policy-based exploration approach in RL and DRL algorithms [114], [209].

6.3.4 State

We consider two types of states; channel state c^s_{t+1}, and DQN algorithm state s_{t+1} [163]. The DQN algorithm state used in the DRL process is based on the channel state. The channel state at time step $t+1$ is $c^s_{t+1} \triangleq (a_t, o_t)$. Hence, the DQN algorithm state at time step $t+1$ is $s_{t+1} \triangleq [c^s_{t-L+2}, \ldots, c^s_t, c^s_{t+1}]$, where L is the state history length that describes the number of past time steps to be tracked by the DQN algorithm.

6.3.5 Reward

Selecting a reward function is usually based on what the RL and DRL systems are supposed to do [3]. First, the selection of the reward function depends on the data specifications, including the length of a sent package and the packet header duration [163]. The larger the packet header duration, the more problems it causes in sending data in the channel (time-delay data transmission or data collision). Therefore, the overhead packet header causes the DQN agent to receive less reward. Afterward, we propose another component to obtain each agent's more precise average reward. Distances between agents may be significant for sending and receiving data and maintaining distances between clusters. Additionally, the distance between two mobile agents is crucial to avoid collisions (the study of mobile agents is beyond the scope of this chapter). Hence, we consider the length of a sent package, the packet's header duration, and the distance between a couple of agents in determining the immediate reward function.

The utilized rewards are:

- At time step t, if the transferred package does not reach the destination (another agent from another cluster or the agent from the same cluster) successfully, and collides on the way, the immediate reward for the i^{th} agent at time step $t+1$ is $r^i_{t+1} = 0$.

- Using the observation set of [163], if the data packet successfully transferred by each agent in the network, the immediate reward for the i^{th} agent at time step $t+1$ is

$$r^i_{t+1} = R^i_c - H^i_p, \tag{6.2}$$

where H_p is the packet's header duration and is a part of each mini-slot.

- By proposing on-time and time-delay observations, we append other component to the immediate reward and present a new immediate reward for each agent. Considering constant $\kappa \in \mathbb{R}^+_*$ and $\kappa \in [1, \infty)$, the new immediate reward for i^{th} agent is introduced by

$$r^i_{t+1} = R^i_c - (\kappa \cdot H^i_p), \tag{6.3}$$

where $\kappa = 1$ if the data packet transferred by each agent in the network successfully and on-time, and $\kappa > 1$ if the data packet transferred by each agent in the network successfully and with time-delay.

- Considering distance between agents who transmit data to each other, we propose another type of immediate reward for the i^{th} agent using the combined immediate reward function [3]. If the data packet successfully transferred by i^{th} agent to j^{th} agent (on-time), we propose the novel distance-based immediate reward for i^{th} agent at time step $t+1$ as

$$r^i_{t+1} = R^i_c - H^i_p - \sum_{j=1}^{N-1} Cr^{ij}_{t+1}. \tag{6.4}$$

If the data packet transferred by i^{th} agent to j^{th} agent successfully and time-delayed, the distance-based immediate reward for i^{th} agent at time step $t+1$ when $\kappa > 1$ is given by

$$r^i_{t+1} = R^i_c - (\kappa \cdot H^i_p) - \sum_{j=1}^{N-1} Cr^{ij}_{t+1}, \tag{6.5}$$

where $Cr_{t+1}^{ij} = \max(Mr_{t+1}^{ij}, Er_{t+1}^{ij}, \check{C}r_{t+1}^{ij}, Nr_{t+1}^{ij})$ is the combined immediate reward function such that Mr_{t+1}^{ij}, Er_{t+1}^{ij}, $\check{C}r_{t+1}^{ij}$, and Nr_{t+1}^{ij} are Manhattan, Euclidean, Chebyshev, and $n-$norm immediate reward functions, respectively [3]. The combined immediate reward function is obtained based on positions (x^i, y^i) and (x^j, y^j) of i^{th} and j^{th} agents, respectively. Nevertheless, the original combined immediate reward function, defined by [3], is based on the current position and the desired position of i^{th} agent in the MARL system.

The formal definition of the DQN target Q-value of a state-action pair (s_t, a_t) is

$$\mathrm{Tar}_Q = R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right), \tag{6.6}$$

using the cumulative reward function $R(s_t, a_t, s_{t+1})$, discount factor $\gamma \in (0,1)$, and target Q-value weight θ^-. According to [163], the DQN target Q-value of a state-action pair (s_t, a_t) is given by

$$\begin{aligned}\mathrm{Tar}_Q &= \frac{r_{t+1} \cdot (1 + \gamma + \ldots + \gamma^{T_d - 1})}{T_d} + \gamma^{T_d} \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right) \\ &= \frac{r_{t+1} \cdot (1 + \gamma + \ldots + \gamma^{T_d - 1})}{T_d} \cdot \frac{1-\gamma}{1-\gamma} + \gamma^{T_d} \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right) \\ &= \frac{r_{t+1}}{T_d} \cdot \frac{1-\gamma^{T_d}}{1-\gamma} + \gamma^{T_d} \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right).\end{aligned} \tag{6.7}$$

The Q-value function $Q(s_t, a_t; \theta)$ is defined by

$$Q(s_t, a_t; \theta) = \mathbb{E}_{s_{t+1} \sim T(s_t, a_t, s_{t+1})} \left[R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right) \right], \tag{6.8}$$

using the state-transition probability function $T\left(s_t, a_t, s_{t+1}\right)$ that defines the conditional probabilities between the states. Furthermore, θ is the Q-value weight. According to the gradient method, the parameter θ is updated as

$$\theta_{t+1} \leftarrow \theta_t + \alpha \left(\mathrm{Tar}_Q - Q(s_t, a_t; \theta)\right) \nabla_\theta Q(s_t, a_t; \theta), \tag{6.9}$$

where α is the learning rate.

Both immediate reward choices (6.2)-(6.5) and the final Q-value function, obtained by updating the parameter θ of (6.9), are connected to the actual data transmission by considering two options: (i) characteristics of transferred packets including the package length and packet header duration; (ii) specifications of neighbouring agents' distances in such a way that unregulated distance between agents delays data transmission.

Remark 6.1 *Learning process in DQN algorithm is more stable than Q-learning process since the update rule introduces a delay between the time when Q-value $Q(s_t, a_t; \theta)$ is updated and the time when target network $Q(s_{t+1}, a_{t+1}; \theta^-)$ is updated [210]. Therefore, the target network remains unchanged due to the time-delay.*

Theorem 6.1 *Suppose that the MAS including N agents is modelled by a graph $\mathscr{G}$, and the learning process is performed by the DQN algorithm. If the i^{th} agent transfers data to the j^{th} agent successfully and with time-delay then the average approximated cumulative team discounted reward of a state-action pair (s_t, a_t) satisfies the following*

$$\frac{1}{N}\underset{\kappa>1}{\sum_{i=1}^{N} Q^i(s_t, a_t; \theta)} < \frac{1}{N}\underset{\kappa=1}{\sum_{i=1}^{N} Q^i(s_t, a_t; \theta)}. \tag{6.10}$$

Proof 6.1 *To avoid time-delay, we consider the action time duration $T_d \in \{1, 2, \ldots, R_{c\max}\}$. With the time-delay, we assume that the action time duration is unbounded above and $T'_d \in \{R_{c\max} + 1, R_{c\max} + 2, \ldots\}$. Therefore, the time-delay occurs when $T'_d \geq R_{c\max} + 1$. We set the constant $\kappa \in \mathbb{R}_*^+$ in such a way that $\kappa \in [1, \infty)$. From (6.3) it follows*

$$\sum_{\substack{i=1 \\ T'^i_d \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}}^{N} \frac{R_c^i - (\kappa \cdot H_p^i)}{T'^i_d} < \sum_{\substack{i=1 \\ T^i_d \in \{1, 2, \ldots, R_{c\max}\}}}^{N} \frac{R_c^i - H_p^i}{T_d^i}. \tag{6.11}$$

Considering a specific value of $\gamma \in (0,1)$, we have

$$\sum_{\substack{i=1 \\ T'^{i}_{d} \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}}^{N} \frac{R_c^i - (\kappa \cdot H_p^i)}{T'^{i}_{d}} \cdot \frac{1-\gamma^{T'^{i}_{d}}}{1-\gamma} < \sum_{\substack{i=1 \\ T^{i}_{d} \in \{1,2,\ldots,R_{c\max}\}}}^{N} \frac{R_c^i - H_p^i}{T_d^i} \cdot \frac{1-\gamma^{T_d^i}}{1-\gamma}. \tag{6.12}$$

Considering a specific value of $\gamma \in (0,1)$, for high values of T'_d the following is given

$$\gamma^{T'_d}_{T'_d \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}} \rightarrow 0 \quad , \tag{6.13}$$

while for $T_d \in \{1,2,\ldots,R_{c\max}\}$ we have

$$0 < \gamma^{T_d}_{T_d \in \{1,2,\ldots,R_{c\max}\}} < 1. \tag{6.14}$$

Using (6.13) and (6.14), for i^{th} agent the following is achieved

$$\sum_{\substack{i=1 \\ T'^{i}_{d} \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}}^{N} \gamma^{T'^{i}_{d}} < \sum_{\substack{i=1 \\ T^{i}_{d} \in \{1,2,\ldots,R_{c\max}\}}}^{N} \gamma^{T^{i}_{d}} . \tag{6.15}$$

Knowing that the target network $Q(s_{t+1}, a_{t+1}; \theta^-) \geq 0$, and according to Remark 6.1, by considering the maximum target network among the possible actions that can be taken from the next state and using (6.15), the following is valid

$$\sum_{\substack{i=1 \\ T'^{i}_{d} \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}}^{N} \gamma^{T'^{i}_{d}} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) < \sum_{\substack{i=1 \\ T^{i}_{d} \in \{1,2,\ldots,R_{c\max}\}}}^{N} \gamma^{T^{i}_{d}} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right). \tag{6.16}$$

Utilizing (6.12) and (6.16) yields

$$\begin{aligned}\underset{T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}{\sum_{i=1}^{N}\Big(\frac{R_c^i-(\kappa\cdot H_p^i)}{T'^{i}_{d}}\cdot\frac{1-\gamma^{T'^{i}_{d}}}{1-\gamma}+\gamma^{T'^{i}_{d}}\max_{a_{t+1}}Q^i\left(s_{t+1},a_{t+1};\theta^-\right)\Big)}\\ <\underset{T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}{\sum_{i=1}^{N}\Big(\frac{R_c^i-H_p^i}{T_d^i}\cdot\frac{1-\gamma^{T_d^i}}{1-\gamma}+\gamma^{T_d^i}\max_{a_{t+1}}Q^i\left(s_{t+1},a_{t+1};\theta^-\right)\Big)}.\end{aligned} \tag{6.17}$$

Therefore,

$$\underset{T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}{\sum_{i=1}^{N}\mathrm{Tar}_{Q^i}} < \underset{T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}{\sum_{i=1}^{N}\mathrm{Tar}_{Q^i}}. \tag{6.18}$$

To achieve the least amount of training loss $\ell(\theta,s,a)$, the difference between the target Q-value and predicted Q-value should converges to zero. Hence, the below equation can be considered for i^{th} agent,

$$\lim_{t\to t_0}Q^i(s_t,a_t;\theta)=\mathrm{Tar}_{Q^i}, \tag{6.19}$$

where t_0 is a certain time. Substituting (6.19) in (6.18) yields

$$\underset{T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}{\sum_{i=1}^{N}\lim_{t\to t_0}Q^i(s_t,a_t;\theta)} < \underset{T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}{\sum_{i=1}^{N}\lim_{t\to t_0}Q^i(s_t,a_t;\theta)}. \tag{6.20}$$

According to the monotone convergence condition, the following is given

$$\sum_{i=1}^{N}\lim_{t\to t_0}Q^i(s_t,a_t;\theta)=\lim_{t\to t_0}\sum_{i=1}^{N}Q^i(s_t,a_t;\theta). \tag{6.21}$$

By considering (6.21), the inequality (6.20) is modified as below

$$\underset{T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}{\lim_{t\to t_0}\sum_{i=1}^{N}Q^i(s_t,a_t;\theta)} < \underset{T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}{\lim_{t\to t_0}\sum_{i=1}^{N}Q^i(s_t,a_t;\theta)} \tag{6.22}$$

$$\sum_{\substack{i=1\\ T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}}^{N} Q^i(s_{t_0},a_{t_0};\theta) \quad < \sum_{\substack{i=1\\ T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}}^{N} Q^i(s_{t_0},a_{t_0};\theta). \tag{6.23}$$

By averaging each side of inequality (6.23), and redistribute each side of the inequality to time t, the following is given

$$\frac{1}{N}\sum_{\substack{i=1\\ T'^{i}_{d}\in\{R_{c\max}+1,R_{c\max}+2,\ldots\}}}^{N} Q^i(s_t,a_t;\theta) \quad < \frac{1}{N}\sum_{\substack{i=1\\ T^{i}_{d}\in\{1,2,\ldots,R_{c\max}\}}}^{N} Q^i(s_t,a_t;\theta). \tag{6.24}$$

$$\begin{cases} T^i_d \in \{1,2,\ldots,R_{c\max}\} & \textit{if} \quad \kappa = 1, \\ T'^{i}_{d} \in \{R_{c\max}+1, R_{c\max}+2,\ldots\} & \textit{if} \quad \kappa > 1. \end{cases} \tag{6.25}$$

Therefore, by considering inequality (6.24) and condition (6.25) for MAS, including N agents, the inequality (6.10) is proven.

Theorem 6.2 *Suppose that graph $\mathscr{G}$ as a MAS includes N agents. The distance between i^{th} agent and j^{th} agent is d^{ij} such that $\xi \le d^{ij} \le \lambda$, where ξ and $\lambda \in \mathbb{R}^+_*$ are constant values and $\xi \ne \lambda$. Using the results of [3] if $\xi \le d^{ij} \le \lambda$, then the distance-based immediate reward (6.5) improves the DQN learning process and compensates for the negative effect of the time-delayed data transmission. Therefore, the average approximated cumulative team discounted reward of a state-action pair (s_t,a_t) satisfies the following*

$$\frac{1}{N}\sum_{\substack{i=1\\ \kappa>1}}^{N} Q^i(s_t,a_t;\theta) \ge \frac{1}{N}\sum_{\substack{i=1\\ \kappa=1}}^{N} Q^i(s_t,a_t;\theta). \tag{6.26}$$

Proof 6.2 *In the case of time-delayed data transmission, distance-based immediate reward, which is calculated based on the distance d^{ij} between i^{th} and j^{th} neighbouring agents, assists the learning process of the DQN agent. Therefore, this immediate reward compensates for the negative effect of time-delayed data transfer at time step t and causes to take more appropriate action at the next time step $t+1$.*

Since the agents are static and the distance d^{ij} between them is constant, the distance-based im-

mediate reward (in combination with the package length and packet header duration) helps the DQN agent to adjust the learning process over time in terms of data transmission speed. Hence, the Q-value is improved at each time step by speeding up the data transmission. This trend will continue in which, at higher time steps, the approximated cumulative team discounted reward of time-delayed data transmission increases more than the on-time data transmission conditions.

6.3.6 DQN Loss

The output layer loss function of the DQN algorithm's NN is the MSE loss function. By decreasing the DQN loss, the DQN reward increases. Therefore, by observing the DQN loss behaviour, the DQN reward performance is predicted. In [211], for uniform action selection, the DQN loss function is given by

$$\ell(\theta,s,a)=\frac{1}{B_e\cdot N}\sum_{i=1}^{N}\sum_{e_t}\left(r_{t+1}^{i}+\gamma\max_{a_{t+1}}Q^{i}\left(s_{t+1},a_{t+1};\theta^{-}\right)-Q^{i}\left(s_t,a_t;\theta\right)\right)^2, \tag{6.27}$$

where B_e is the experience replay mini-batch size. Using the non-uniform action (6.1), containing the set of $R_c \in \{0,1,2,\ldots,R_{c\max}\}$ as possible actions, and applying target Q-value (6.7) as well as predicted Q-value, the DQN loss function for non-uniform action selection with action time duration $T_d \in \{1,2,\ldots,R_{c\max}\}$ is defined as

$$\ell(\theta,s,a)=\frac{1}{B_e\cdot N}\sum_{i=1}^{N}\sum_{\substack{e_t\\ T_d\in\{1,2,\ldots,R_{c\max}\}}}\left(\frac{r_{t+1}^{i}}{T_d}\cdot\frac{1-\gamma^{T_d}}{1-\gamma}+\gamma^{T_d}\max_{a_{t+1}}Q^{i}\left(s_{t+1},a_{t+1};\theta^{-}\right)-Q^{i}\left(s_t,a_t;\theta\right)\right)^2, \tag{6.28}$$

where $e_t=(s_t,a_t,T_d,r_{t+1},s_{t+1})$ is the experience at time t that is the simplified version of

$$e_t=(c_t^s,a_t,T_d,r_{t+1},c_{t+1}^s).$$

Note that (6.27) is derived from (6.28) if $T_d = 1$. Moreover, the time-delayed DQN loss function for action time duration $T'_d \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}$ is given by

$$\ell(\theta, s, a) = \frac{1}{B_e \cdot N} \sum_{\substack{i=1 \\ T'_d \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}}^{N} \sum_{e'_t} \Big(\frac{r^i_{t+1}}{T'_d} \cdot \frac{1-\gamma^{T'_d}}{1-\gamma} + \gamma^{T'_d} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) - Q^i\left(s_t, a_t; \theta\right) \Big)^2, \quad (6.29)$$

for experience $e'_t = (s_t, a_t, T'_d, r_{t+1}, s_{t+1})$ at time t that is achieved from $e'_t = (c^s_t, a_t, T'_d, r_{t+1}, c^s_{t+1})$. Note that average loss calculations based on experience e_t and experience e'_t are performed from $m = 1$ to B_e as the experience replay mini-batch size.

6.3.7 Adversarial Attacks

Three types of gradient-based adversarial attacks are considered to benchmark the data transmission robustness of the proposed leader-follower MAS by considering the new observation set and the proposed distance-based immediate reward (Figure 6.2). The changes made by this type of attacks are very subtle, but they can also affect the system's performance. Before occurring an attack, the DQN algorithm aims to reduce the average training loss in a given time step and enhance the average reward.

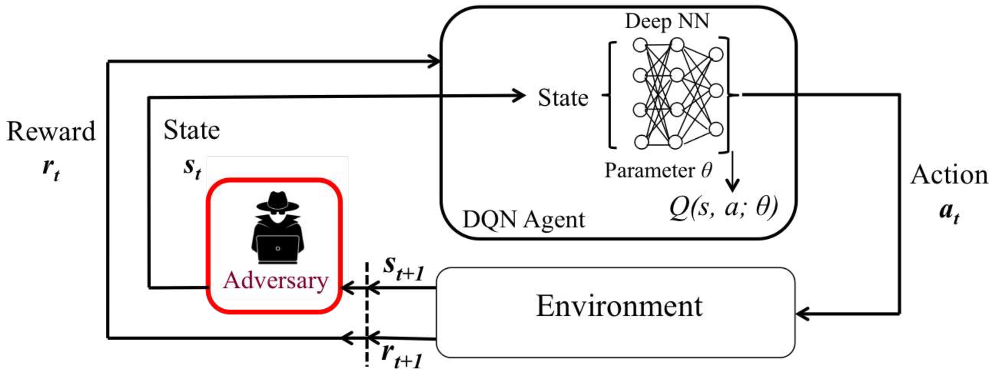


Figure 6.2: A DQN agent's structure affected by an adversarial attack.

6.3.7.1 FGSM Adversarial Attack

The FGSM is a type of attack proposed in [169]. Our methodology involves attacking the system's state by causing the FGSM adversary to make very few changes to the state over a period of time to increase the system's average training loss. Using the gradient of the loss function with respect to the state, FGSM maximizes the perturbation and minimizes the difference between the perturbed and original inputs [169], [173], [212]. In this regard, using (6.28) and (6.29), for on-time and time-delayed data transmission, respectively, the FGSM attack signal (perturbation) is obtained by

$$\eta = \varepsilon \cdot \mathrm{sign}\left(\nabla_s \ell(\theta, s, a)\right), \tag{6.30}$$

where ε is the attack magnitude to ensure the perturbations are small, and sign$(.)$ is the sign function. Further, ∇_s is the gradient of the loss function related to model state s as well as correct action a, ℓ is the loss function of DQN agent, and θ is the model parameters. After adding the attack signal to the state s, the adversarial input s_{adv} is calculated as follows

$$\begin{aligned} s_{adv} &= s + \eta \\ &= s + \varepsilon \cdot \mathrm{sign}\left(\nabla_s \ell(\theta, s, a)\right), \end{aligned} \tag{6.31}$$

where L_∞-norm bound $\|\eta\|_\infty \leq \varepsilon$ for perturbation η. Using (6.29) and (6.31), the adversarial input s_{adv} for time-delayed data transmission is given by

$$s_{adv} = s + \varepsilon \cdot \mathrm{sign}\left(\nabla_s \frac{1}{B_e \cdot N} \sum_{i=1}^{N} \sum_{\substack{e'_t \\ T'_d \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}}} \left(\frac{r^i_{t+1}}{T'_d} \cdot \frac{1-\gamma^{T'_d}}{1-\gamma} + \gamma^{T'_d} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) - Q^i\left(s_t, a_t; \theta\right)\right)^2\right). \tag{6.32}$$

Once s_{adv} is calculated, it is fed to the NN and replaces the primary input s of the NN. The NN is fooled and trained based on the adversarial input s_{adv}.

6.3.7.2 FGM Adversarial Attack

The FGM attack signal is a generalization of FGSM attack signal and is calculated as:

$$\eta = \varepsilon \cdot \frac{\nabla_s \ell(\theta, s, a)}{\|\nabla_s \ell(\theta, s, a)\|_2}. \tag{6.33}$$

Using (6.33), the adversarial input s_{adv} is calculated by

$$\begin{aligned} s_{adv} &= s + \eta \\ &= s + \varepsilon \cdot \frac{\nabla_s \ell(\theta, s, a)}{\|\nabla_s \ell(\theta, s, a)\|_2}, \end{aligned} \tag{6.34}$$

where L_2-norm bound $\|\eta\|_2 \leq \varepsilon$ for perturbation η. By substituting (6.29) in (6.34), the adversarial input s_{adv} for time-delayed data transmission is given. The training procedure is performed similarly to the FGSM adversarial attack.

6.3.7.3 BIM Adversarial Attack

The BIM attack is a simple and straight extension of the FGSM attack proposed by [172]. This method uses a fast gradient multiple times by considering small step size instead of applying the perturbation in a single step. The BIM attack signal and the adversarial input s_{adv} is given by

$$\eta = \beta \cdot \operatorname{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a)\right), \tag{6.35}$$

$$\begin{aligned} s_{t+1}^{adv} &= s_t^{adv} + \eta \\ &= s_t^{adv} + \beta \cdot \operatorname{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a)\right), \end{aligned} \tag{6.36}$$

where $\beta = \frac{\varepsilon}{T}$ is a small step size and T is the number of iterations. Using (6.29) in (6.36), the adversarial input s_{t+1}^{adv} for time-delayed data transmission is provided.

6.3.8 First Adversarial Attack Defence

We provide a simple but effective approach to defend against adversarial attacks and mitigate their destructive effects on the MADRL system performance (Figure 6.3).

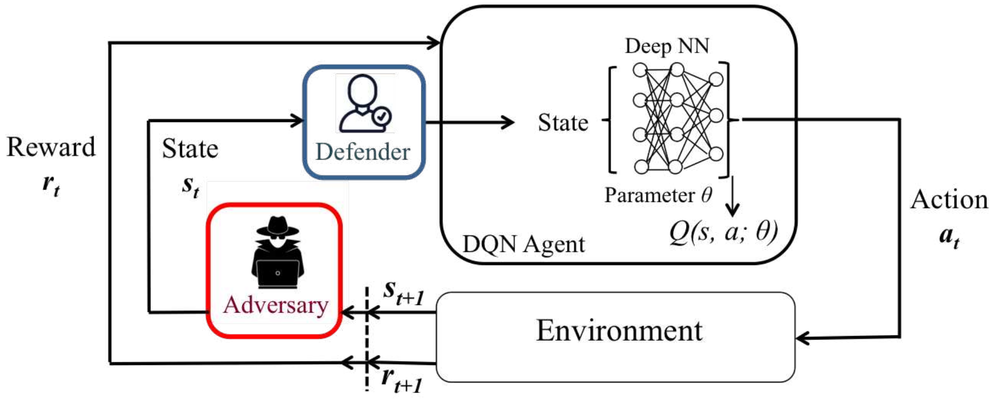


Figure 6.3: A DQN agent's structure affected by an adversarial attack and defence algorithm.

Algorithm 6.1 First Adversarial Attack Defence

```
    Input: $s$, $s_t^{adv}$, $\theta$, $T$, $T_{max}$, $\varepsilon$
    Output: $s_{adv}$, $s_{t+1}^{adv}$
  for $T = 0, 1, 2, \ldots, T_{max}$ do
    if $attack = FGSM$ then
      $\eta = \varepsilon \cdot \text{sign}\left(\nabla_s \ell(\theta, s, a)\right)$;
    else if $attack = FGM$ then
      $\eta = \varepsilon \cdot \frac{\nabla_s \ell(\theta,s,a)}{\|\nabla_s \ell(\theta,s,a)\|_2}$;
    else if $attack = BIM$ then
      $\beta = \frac{\varepsilon}{T}$,
      $\eta = \beta \cdot \text{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a)\right)$;
    end if
    while $attack = True$ do
      if $attack = FGSM$ OR $attack = FGM$ then
        $s^* = \arg\max_{s}(\eta)$,
        $\eta \longleftarrow s^*$,
        $s_{adv} = s + \eta$;
      else if $attack = BIM$ then
        $s^* = \arg\max_{s_t^{adv}}(\eta)$,
        $\eta \longleftarrow s^*$,
        $s_{t+1}^{adv} = s_t^{adv} + \eta$.
      end if
    end while
  end for
```

In the proposed Algorithm 6.1 that is based on NN behaviour, we consider the `argmax` operation on perturbation vector η to find the argument that gives the maximum value from η. In other words, we desire to find a state that provides the maximum perturbation value. We assume that the FGSM, FGM, or BIM adversarial attacks are detectable. Once one of the FGSM, FGM, or BIM adversarial attacks is detected, the state vector s^* is calculated based on the set of states (inputs) and substituted with perturbation vector η as follows

$$\underset{n\times 1}{\vec{s^*}} = \arg\max_{s}(\underset{n\times 1}{\vec{\eta}}), \tag{6.37}$$

$$\underset{n\times 1}{\vec{\eta}} \longleftarrow \underset{n\times 1}{\vec{s^*}}, \tag{6.38}$$

where η and s^* are $n \times 1$ vectors. The state vector s^*, which determines maximum perturbation value, has the worst effect on the MAS performance; however, the DNN learns from the state vector s^* and uses its negative feedback to improve the system performance during an adversarial attack. For BIM adversarial attack the Equation (6.37) is presented as $\underset{n\times 1}{\vec{s^*}} = \arg\max_{s_t^{adv}}(\underset{n\times 1}{\vec{\eta}})$.

6.3.8.1 FGSM Adversarial Attack Defence

Using (6.30) and (6.31), the state vector s^* and the adversarial input s_{adv} are calculated to defend against the FGSM adversarial attack as follows

$$s^* = \arg\max_{s}(\varepsilon \cdot \text{sign}(\nabla_s \ell(\theta, s, a))), \tag{6.39}$$

$$\varepsilon \cdot \text{sign}\,(\nabla_s \ell(\theta, s, a)) \longleftarrow s^*, \tag{6.40}$$

$$\begin{aligned} s_{adv} &= s + \eta \\ &= s + \varepsilon \cdot \text{sign}\,(\nabla_s \ell(\theta, s, a)) \\ &= s + s^*. \end{aligned} \tag{6.41}$$

6.3.8.2 FGM Adversarial Attack Defence

Utilizing (6.33) and (6.34), the state vector s^* and the adversarial input s_{adv} are computed to defend against the FGM adversarial attack as below

$$s^* = \arg\max_{s} \left(\varepsilon \cdot \frac{\nabla_s \ell(\theta, s, a)}{\|\nabla_s \ell(\theta, s, a)\|_2} \right), \tag{6.42}$$

$$\varepsilon \cdot \frac{\nabla_s \ell(\theta, s, a)}{\|\nabla_s \ell(\theta, s, a)\|_2} \longleftarrow s^*, \tag{6.43}$$

$$\begin{aligned} s_{adv} &= s + \eta \\ &= s + \varepsilon \cdot \frac{\nabla_s \ell(\theta, s, a)}{\|\nabla_s \ell(\theta, s, a)\|_2} \\ &= s + s^*. \end{aligned} \tag{6.44}$$

6.3.8.3 BIM Adversarial Attack Defence

Using (6.35) and (6.36), the state vector s^* and the adversarial input s_{t+1}^{adv} are calculated to defend against the BIM adversarial attack as follows

$$s^* = \arg\max_{s_t^{adv}} \left(\beta \cdot \text{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a) \right) \right), \tag{6.45}$$

$$\beta \cdot \text{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a) \right) \longleftarrow s^*, \tag{6.46}$$

$$\begin{aligned} s_{t+1}^{adv} &= s_t^{adv} + \eta \\ &= s_t^{adv} + \beta \cdot \text{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a) \right) \\ &= s_t^{adv} + s^*. \end{aligned} \tag{6.47}$$

Algorithm 6.2 Second Adversarial Attack Defence

```
    Input: $s$, $s_t^{adv}$, $\theta$, $T$, $T_{max}$, $\varepsilon$
    Output: $s_{adv}$, $s_{t+1}^{adv}$
  for $T = 0, 1, 2, \dots, T_{max}$ do
      if $attack = FGSM$ then
          $\eta = \varepsilon \cdot \text{sign}\left(\nabla_s \ell(\theta, s, a)\right)$;
      else if $attack = FGM$ then
          $\eta = \varepsilon \cdot \frac{\nabla_s \ell(\theta,s,a)}{\|\nabla_s \ell(\theta,s,a)\|_2}$;
      else if $attack = BIM$ then
          $\beta = \frac{\varepsilon}{T}$,
          $\eta = \beta \cdot \text{sign}\left(\nabla_{s_t^{adv}} \ell(\theta, s_t^{adv}, a)\right)$;
      end if
      while $attack = True$ do
          if $attack = FGSM$ OR $attack = FGM$ then
              function convert($s$)
                  return $-s$

              end function
              $s \longleftarrow -s$,
              attacker generates $\eta$,
              $s^* = \arg\max_s(\eta)$,
              $\eta \longleftarrow s^*$,
              $s_{adv} = s + \eta$;
          else if $attack = BIM$ then
              function convert($s_t^{adv}$)
                  return $-s_t^{adv}$

              end function
              $s_t^{adv} \longleftarrow -s_t^{adv}$,
              attacker generates $\eta$,
              $s^* = \arg\max_{s_t^{adv}}(\eta)$,
              $\eta \longleftarrow s^*$,
              $s_{t+1}^{adv} = s_t^{adv} + \eta$.
          end if
      end while
  end for
```

6.3.9 Second Adversarial Attack Defence

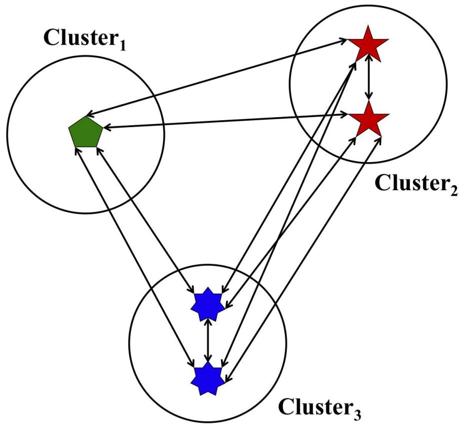
We provide another effective method to mitigate the gradient-based attacks' destructive effects on the MADRL system performance and defend against the discussed adversarial attacks. We assume that the FGSM, FGM, or BIM adversarial attacks are detectable. In Algorithm 6.2, that is an extension of Algorithm 6.1, once one of the FGSM, FGM, or BIM adversarial attacks is detected, a `convert` function changes the sign of the state. Hence, before the attacker can confuse the NN, the state is modified and replaced with the correct state that was fed to the NN. This is done to mislead the attacker, so that the attacker generates the attack signal η based on the converted state. Changing the state sign not only fools the attacker and reduces its destructive effects but also causes the generated attack signal by the attacker to be used for appropriate NN training. The remain of the Algorithm 6.2 performs similar to the Algorithm 6.1.

Some application domains for Sections 6.3.8 and 6.3.9 are attack detection by various antivirus software and provide security for e-mail contents, E-commerce, streaming media, databases, webs, file transfer protocol (FTP) servers, etc.

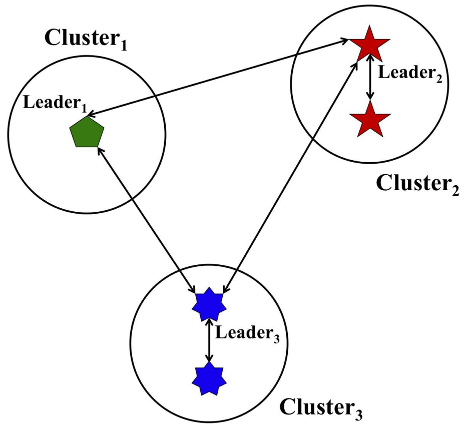
6.4 Results and Discussion

We illustrate results for on-time and time-delayed data transmission between agents of heterogeneous MAS with and without a leader, using the DQN algorithm. Additionally, the impacts of FGSM, FGM, and BIM attacks, as well as the consequences of the defence algorithms on the proposed leader-follower system, are illustrated and shown numerically.

Two types of graphs $\mathscr{G}$ are considered: complete (leaderless) and connected (leader-follower) graphs. The leaderless and leader-follower scenarios, including $N = 5$ static, heterogeneous agents, and $P = 3$ clusters, are illustrated in Figure 6.4.



(a) Communication topology of a complete graph $\mathscr{G}$ without any leader.



(b) Communication topology of a connected graph $\mathscr{G}$ with a leader at each cluster.

Figure 6.4: Two heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters.

The adjacency matrices of the leaderless (left) and leader-follower (right) MAS are given by

$$\underset{5\times5}{A_{\mathscr{G}}} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad \underset{5\times5}{A_{\mathscr{G}}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

The degree matrices of the leaderless (left) and leader-follower (right) MAS are

$$\underset{5\times5}{D_{\mathscr{G}}} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix}, \quad \underset{5\times5}{D_{\mathscr{G}}} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

In both types of graphs, the DQN agent's internal structure consists of feed-forward NN architecture for training, including 36 layers with Adam optimizer and MSE loss function. Note that we have used the trial and error method to choose the number of NN hidden layers. To select any number of layers, we performed the learning process five times to reach a definite result about the number of layers. The activation function of all 36 layers is ReLU function. The DQN agent learning rate is $\alpha = 0.01$, the discount factor is $\gamma = 0.999$, the experience replay mini-batch size is $B_e = 64$, and the constant positive real number to calculate immediate reward r^i_{t+1} is $\kappa = 4$ if the data packet is transferred in the network with time-delay. The packet's header duration for all agents is considered as $H_p = 0.5$. The threshold to determine the on-time or time-delayed data transmission is 11 mini-slots. To compute the attack signal, the attack magnitude is $\varepsilon = 0.6$, and the number of iterations is $T_{max} = 30000$. The five agents' two-dimensional positions are

$$(x^i, y^i) = \{(0.1, 0.22), (0.3, 0.27), (0.21, 0.9), (0.3, 0.23), (0.2, 0.4)\},$$

where $i \in \{1, 2, ..., 5\}$. The positions, which are used to obtain the distance-based immediate rewards of (6.4) and (6.5), remain constant during the total time steps due to the static agents. As opposed to this, when agents are mobile, their positions should be updated and added to the list of former positions at any time step, as we will investigate in the subsequent research. Moreover, the returned values of the novel observation set are

$$\begin{cases} [0,0,0,0,1] = busy, \\ [0,0,0,1,0] = idle, \\ [0,0,1,0,0] = on-time, \\ [0,1,0,0,0] = time-delay, \\ [1,0,0,0,0] = collided. \end{cases} \tag{6.48}$$

All scenarios are carried out during the 20000 time steps for the data transmission part of the experiment. The experiments are performed during the 30000 time steps while investigating the data transmission robustness due to various adversarial attacks. The results are shown after five times training to ensure the reliability of the results.

We have used, modified, and extended a part of the code given in [213] as a part of our implementation. Furthermore, for algorithm's execution, a system with 3.60 GHz Intel Core i7 − 7700

Table 6.1: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps without considering the distance-based reward.

Various Graphs \ **Rewards and Loss**	**Agent 1 Reward**	**Agent 2 Reward**	**Agent 3 Reward**	**Agent 4 Reward**	**Agent 5 Reward**	**Team Reward**	**DQN Loss**
Leaderless MAS	0.0047	0.0332	0.0665	0.1330	0.0997	0.3372	3411.58
Leaderless MAS with time-delay	0.0227	0.0427	0.0427	0.0380	0.0997	0.2460	6924.71
Leader-follower MAS	0.0147	0.0475	0.0570	0.0760	0.0902	0.2855	7325.22
Leader-follower MAS with time-delay	0.0340	0.0475	0.0237	0.0617	0.0902	0.2572	8400.12

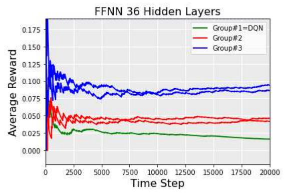
Table 6.2: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps by considering the novel distance-based reward.

Various Graphs \ **Rewards and Loss**	**Agent 1 Reward**	**Agent 2 Reward**	**Agent 3 Reward**	**Agent 4 Reward**	**Agent 5 Reward**	**Team Reward**	**DQN Loss**
Leaderless MAS	0.0202	0.0438	0.0485	0.0930	0.0735	0.2792	6037.81
Leaderless MAS with time-delay	0.0290	0.0438	0.0533	0.1028	0.1275	0.3566	6493.40
Leader-follower MAS	0.0095	0.0674	0.0288	0.0671	0.0768	0.2498	4116.27
Leader-follower MAS with time-delay	0.0322	0.0626	0.0336	0.0959	0.1008	0.3252	7983.38

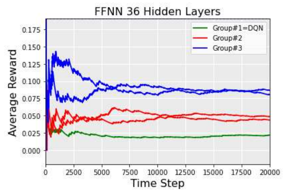
processor, 16 GB installed RAM, 64−bit operating system, and x64−based processor is used.

6.4.1 Multi-agent Performance Analysis

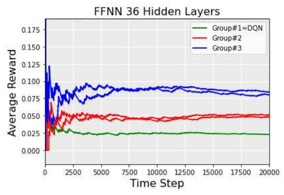
According to Table 6.1, without considering distance-based reward and time-delay, both leaderless and leader-follower MAS scenarios achieve the superior team reward compared to the case when the packets transfer in the network with time-delay. In this case and for leaderless MAS, by considering time-delay, the team reward has been reduced by -27.04%. In a similar situation and for leader-follower MAS, by considering time-delay, the team reward has been decreased by -9.91%. Figure 6.5 illustrates the reward convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps for leaderless and leader-follower scenarios by considering on-time and time-delay observations. Based on the results in Table 6.1 and Figure 6.5, delay in sending data reduces team rewards for both leaderless and leader-follower scenarios.



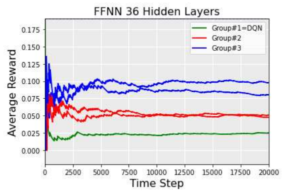
(a) Leaderless MAS.



(b) Leaderless MAS by considering time-delay.

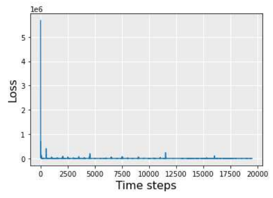


(c) Leader-follower MAS.

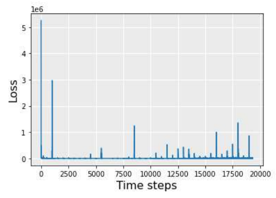


(d) Leader-follower MAS by considering time-delay.

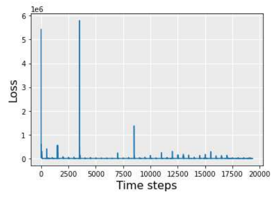
Figure 6.5: Reward convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps.



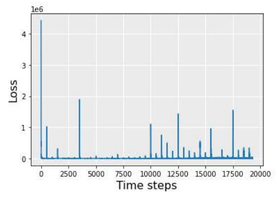
(a) Leaderless MAS.



(b) Leaderless MAS by considering time-delay.

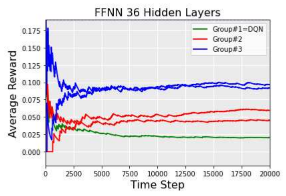


(c) Leader-follower MAS.

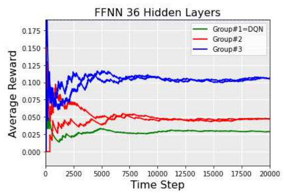


(d) Leader-follower MAS by considering time-delay.

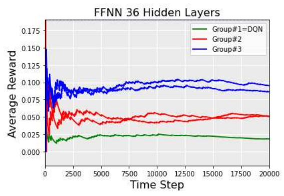
Figure 6.6: Loss convergence of the DQN algorithm in a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps.



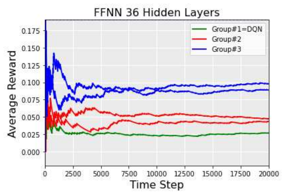
(a) Leaderless MAS.



(b) Leaderless MAS by considering time-delay.

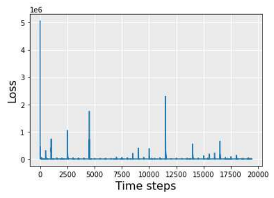


(c) Leader-follower MAS.

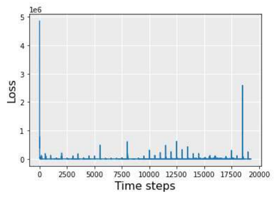


(d) Leader-follower MAS by considering time-delay.

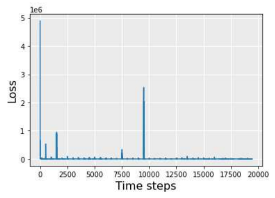
Figure 6.7: Reward convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps by considering the novel distance-based reward.



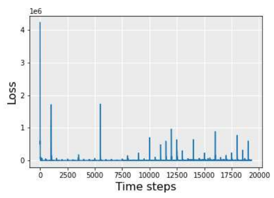
(a) Leaderless MAS.



(b) Leaderless MAS by considering time-delay.



(c) Leader-follower MAS.



(d) Leader-follower MAS by considering time-delay.

Figure 6.8: Loss convergence of the DQN algorithm in a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps by considering the novel distance-based reward.

As can be seen from Table 6.1 and Table 6.2, the simulation confirms the claim of Theorem 6.2 in such a way that the distance-based immediate reward has improved the system performance despite the time-delayed data transmission (regardless of whether the system is leaderless or leader-follower). Moreover, according to Table 6.1, without considering distance-based reward and time-delay, the DQN algorithm in both leaderless and leader-follower MAS scenarios achieves the less average loss compared to the case when the packets transfer in the network with time-delay. For leaderless MAS, by considering time-delay, the average DQN loss has been increased by $+102.97\%$. For leader-follower MAS, by considering time-delay, the average DQN loss has been enhanced by $+14.67\%$. Figure 6.6 shows the DQN loss convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps for leaderless and leader-follower scenarios by considering on-time and time-delay observations. The large fluctuations in the amount of loss after 10000 time steps in Figures 6.6b and 6.6d are due to delay in data transmission.

Considering distance-based reward and time-delay, both leaderless and leader-follower MAS scenarios achieve the higher team reward compared to the criteria when the packets transfer in the network on-time (Table 6.2). In this case and for leaderless MAS, by considering time-delay, the team reward has been increased by $+27.72\%$. In a comparable status and for leader-follower MAS, by considering time-delay, the team reward has been enhanced by $+30.18\%$. Figure 6.7 shows the reward convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps for leaderless and leader-follower scenarios by considering on-time and time-delay observations as well as distance-based reward. Based on the results in Table 6.2 and Figure 6.7, the proposed distance-based immediate reward, in combination with the previous immediate reward, covers the negative effects of data transmission delays for both leaderless and leader-follower topologies.

Considering distance-based reward and time-delay, the DQN algorithm in leaderless and leader-follower MAS scenarios achieves the higher loss compared to the case when the packets transfer in the network on-time (Table 6.2). For leaderless MAS, by considering time-delay, the average DQN loss has been increased by $+7.54\%$. In a similar criteria and for leader-follower MAS, by considering time-delay, the average DQN loss has been enhanced by $+93.94\%$. Figure 6.8 demonstrates the DQN loss convergence of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 20000 time steps for leaderless and leader-follower scenarios by considering on-time and time-delay observations as well as distance-based reward. The time-delayed data transmission has caused the large fluctuations in the amount of loss after 10000 time steps in Figures 6.8b and 6.8d. As can be seen from Table 6.1 and Table 6.2, in scenarios that data is transmitted with time-delay, the average loss of DQN is increased compared to the cases where data is transferred on-time.

Note that the percentage increase of average loss and reward are calculated by

$$\% \, Inc = 100 \times \frac{(\text{With Time-Delay - Without Time-Delay})}{|\text{ Without Time-Delay }|}. \tag{6.49}$$

Moreover, the percentage decrease of average loss and reward are computed by

$$\% \, Dec = 100 \times \frac{(\text{Without Time-Delay - With Time-Delay})}{|\text{ Without Time-Delay }|}. \tag{6.50}$$

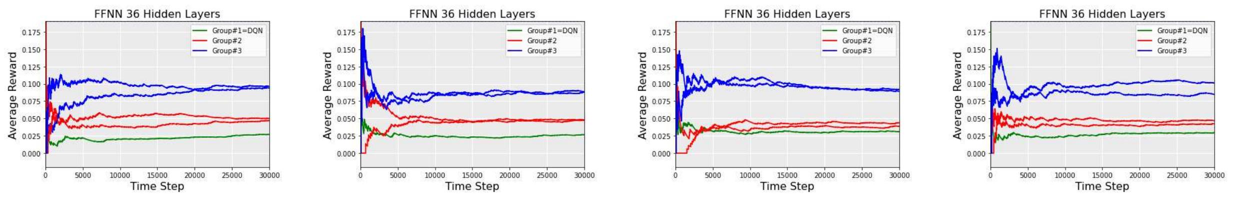
6.4.2 Performance Analysis of the Proposed MAS Under Adversarial Attacks

According to Table 6.3, by considering time-delayed data transmission and distance-based reward, the team reward of the leader-follower MADRL system including $N = 5$ agents in $P = 3$ various clusters without adversarial attack equals to 0.3415 (Figure 6.9a). Moreover, in similar conditions, the DQN loss of the discussed MADRL system is 6996.28 (Figure 6.10a). Under FGSM adversarial attack, the team reward of the leader-follower MADRL system is decreased to 0.3236 by -5.24% (Figure 6.9b), and the DQN loss is increased to 20920.10 by $+199.01\%$ (Figure 6.10b). Furthermore, under FGM adversarial attack, the MADRL system team reward is reduced to 0.3054 by -10.57% (Figure 6.9c), and the DQN loss is enhanced to 60232.71 by

+760.92% (Figure 6.10c). Under BIM adversarial attack, the team reward of the leader-follower MADRL system is declined to 0.2929 by -14.23% (Figure 6.9d), and the DQN loss is increased to 27949.57 by +299.49% (Figure 6.10d). Hence, it is evident that the time-delayed data transmission of the proposed leader-follower MADRL system is not robust under three types of adversarial attacks during 30000 time steps, meaning that its team reward is reduced after attack, and the DQN loss is enhanced.

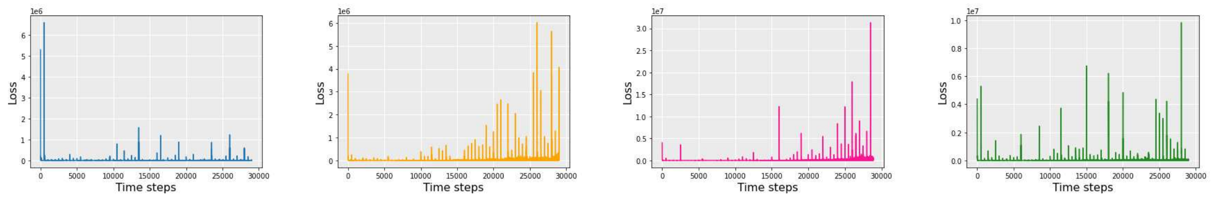
Table 6.3: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward under FGSM, FGM, and BIM adversarial attacks.

Rewards and Loss / **Various Attacks**	**Agent 1 Reward**	**Agent 2 Reward**	**Agent 3 Reward**	**Agent 4 Reward**	**Agent 5 Reward**	**Team Reward**	**DQN Loss**
Leader-follower MAS without attack	0.0340	0.0674	0.0432	0.0911	0.1056	0.3415	6996.28
Leader-follower MAS with FGSM attack	0.0353	0.0626	0.0432	0.0767	0.1056	0.3236	20920.10
Leader-follower MAS with FGM attack	0.0315	0.0771	0.0480	0.0719	0.0768	0.3054	60232.71
Leader-follower MAS with BIM attack	0.0335	0.0385	0.0480	0.0719	0.1008	0.2929	27949.57



(a) Without attack. (b) Under FGSM attack. (c) Under FGM attack. (d) Under BIM attack.

Figure 6.9: Reward convergence of a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward under FGSM, FGM, and BIM adversarial attacks.

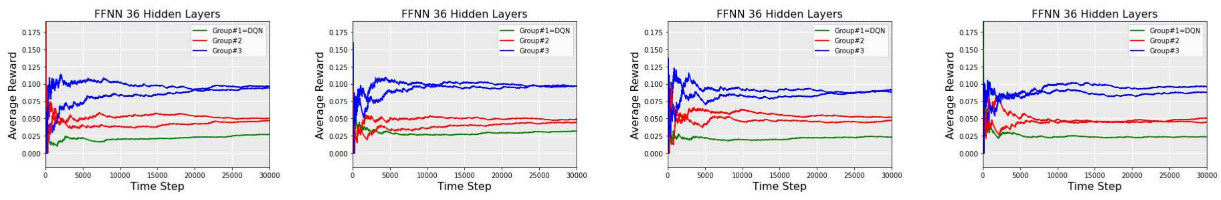


(a) Without attack. (b) Under FGSM attack. (c) Under FGM attack. (d) Under BIM attack.

Figure 6.10: Loss convergence of the DQN algorithm in a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward under FGSM, FGM, and BIM adversarial attacks.

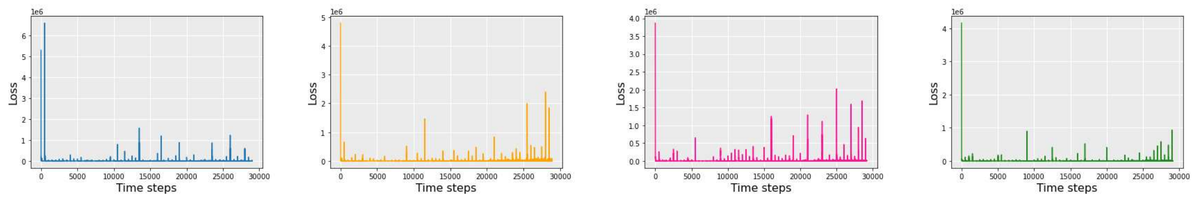
6.4.3 Performance Analysis of the Proposed MAS After Applying First Adversarial Attack Defence

According to Tables 6.3 and 6.4, after using the proposed adversarial attack defence Algorithm 6.1, the destructive effects of the FGSM, FGM, and BIM malicious attacks are mitigated during 30000 time steps. In this regard, the team reward of the leader-follower MADRL system is reached 0.3433 from 0.3236 by $+6.08\%$ after applying the adversarial attack defence method against the FGSM attack (Figure 6.11b). Moreover, the DQN loss is decreased from 20920.10 to 8081.50 by -61.36% (Figure 6.12b). For FGM adversarial attack and after using the introduced defence procedure, the team reward of the MADRL system is enhanced from 0.3054 to 0.3342 by $+9.43\%$ (Figure 6.11c). The DQN loss is reduced from 60232.71 to 6966.24 by -88.43% (Figure 6.12c). Furthermore, the team reward of the MADRL system under BIM attack is enhanced from 0.2929 to 0.3336 by $+13.89\%$, and the DQN loss is decreased from 27949.57 to 3705.51 by -86.74% after utilizing the suggested attack defence technique (Figures 6.11d and 6.12d).



(a) Without attack. (b) After defend against FGSM attack. (c) After defend against FGM attack. (d) After defend against BIM attack.

Figure 6.11: Reward convergence of a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.1.



(a) Without attack. (b) After defend against FGSM attack. (c) After defend against FGM attack. (d) After defend against BIM attack.

Figure 6.12: Loss convergence of the DQN algorithm in a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.1.

Table 6.4: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.1.

Rewards and Loss / Various Attacks	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
Leader-follower MAS without attack	0.0340	0.0674	0.0432	0.0911	0.1056	0.3415	6996.28
Leader-follower MAS with FGSM attack	0.0359	0.0626	0.0480	0.1150	0.0815	0.3433	8081.50
Leader-follower MAS with FGM attack	0.0172	0.0433	0.0720	0.1294	0.0720	0.3342	6966.24
Leader-follower MAS with BIM attack	0.0213	0.0771	0.0384	0.0959	0.1008	0.3336	3705.51

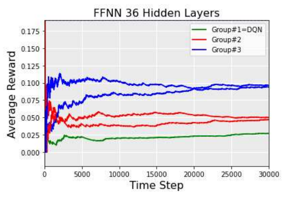
6.4.4 Performance Analysis of the Proposed MAS After Applying Second Adversarial Attack Defence

According to Table 6.3 and Table 6.5, after using the proposed adversarial attack defence Algorithm 6.2, the destructive effects of the FGSM, FGM, and BIM malicious attacks are mitigated during 30000 time steps. The team reward of the leader-follower MADRL system is reached 0.3563 from 0.3236 by $+10.10\%$ after applying the adversarial attack defence method against the FGSM attack (Figure 6.13b). Moreover, the DQN loss is decreased from 20920.10 to 2905.81 by -86.11% (Figure 6.14b). For FGM adversarial attack and after using the introduced defence procedure, the team reward of the MADRL system is enhanced from 0.3054 to 0.3187 by $+4.35\%$ (Figure 6.13c). The DQN loss is reduced from 60232.71 to 4100.37 by -93.19% (Figure 6.14c). Furthermore, the team reward of the MADRL system under BIM attack is enhanced from 0.2929 to 0.3292 by $+12.39\%$, and the DQN loss is decreased from 27949.57 to 4526.58 by -83.80% after utilizing the suggested attack defence technique (Figures 6.13d and 6.14d).

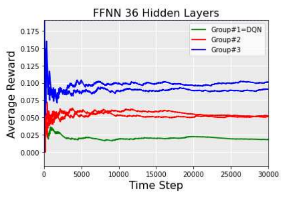
Figures 6.15a and 6.15b show the team reward and DQN loss before and after defence Algorithm 6.1 against various adversarial attacks, respectively. Figures 6.16a and 6.16b show the team reward and DQN loss before and after defence Algorithm 6.2 against various adversarial attacks, respectively.

Table 6.5: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.2.

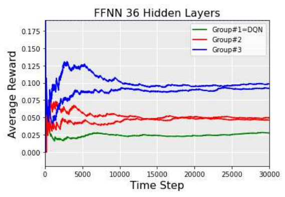
Rewards and Loss / **Various Attacks**	**Agent 1 Reward**	**Agent 2 Reward**	**Agent 3 Reward**	**Agent 4 Reward**	**Agent 5 Reward**	**Team Reward**	**DQN Loss**
Leader-follower MAS without attack	0.0340	0.0674	0.0432	0.0911	0.1056	0.3415	6996.28
Leader-follower MAS with FGSM attack	0.0105	0.0626	0.0480	0.1246	0.1103	0.3563	2905.81
Leader-follower MAS with FGM attack	0.0306	0.0578	0.0288	0.1006	0.1007	0.3187	4100.37
Leader-follower MAS with BIM attack	0.0314	0.0530	0.0528	0.1006	0.0911	0.3292	4526.58



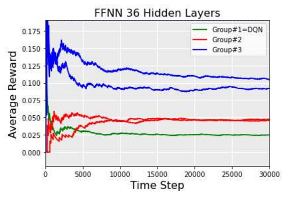
(a) Without attack.



(b) After defend against FGSM attack.

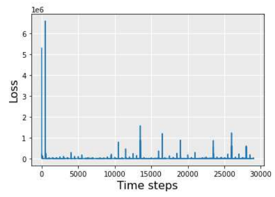


(c) After defend against FGM attack.

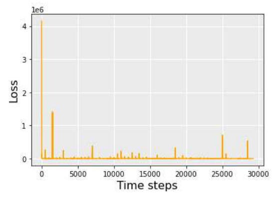


(d) After defend against BIM attack.

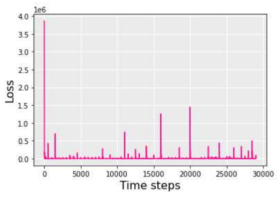
Figure 6.13: Reward convergence of a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.2.



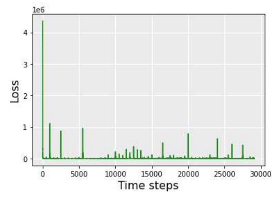
(a) Without attack.



(b) After defend against FGSM attack.



(c) After defend against FGM attack.

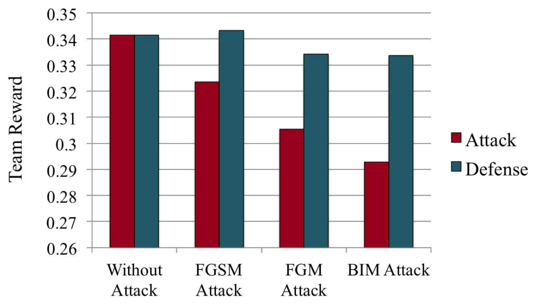


(d) After defend against BIM attack.

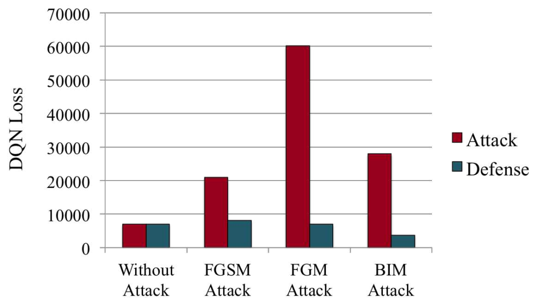
Figure 6.14: Loss convergence of the DQN algorithm in a heterogeneous leader-follower MAS, including $N = 5$ agents in $P = 3$ different clusters, during 30000 time steps by considering time-delay and distance-based reward after adversarial attack defence Algorithm 6.2.

6.4.5 Variety of Agents

According to [163], in the proposed topology, there are three types of agents assigned to $P = 3$ different clusters. The first cluster's agents use a DQN architecture. The agents in the second cluster follow the ALOHA protocol [214], [215], [216]. Moreover, the time division multiple access (TDMA) protocol makes up the agents' architecture of the third cluster [217]. The concentration of this chapter is on DQN agent behaviours (first cluster) and their effects on the other clusters of

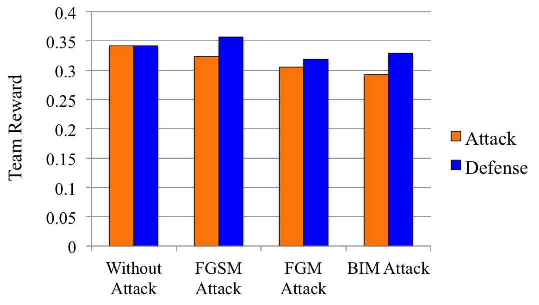


(a) Team reward before and after defence Algorithm 6.1.

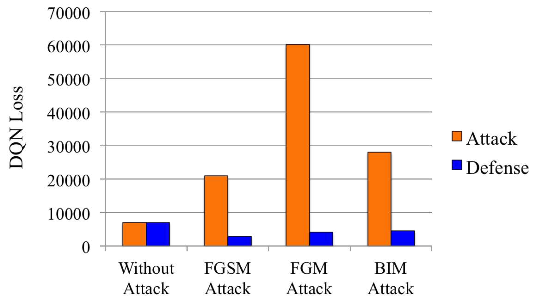


(b) DQN loss before and after defence Algorithm 6.1.

Figure 6.15: Team reward and DQN loss before and after using defence Algorithm 6.1 against various adversarial attacks, during 30000 time steps.



(a) Team reward before and after defence Algorithm 6.2.



(b) DQN loss before and after defence Algorithm 6.2.

Figure 6.16: Team reward and DQN loss before and after using defence Algorithm 6.2 against various adversarial attacks, during 30000 time steps.

agents' performance of the MADRL system in different situations.

Note that for developing and simulating this chapter's algorithms, Python Programming Language is utilized.

6.5 Conclusions

We studied the on-time and time-delayed data transmission of a leaderless (complete graph), heterogeneous, MADRL system using the DQN algorithm. Moreover, we investigated the on-time and time-delayed data transmission of a leader-follower (connected graph), heterogeneous, MADRL system using the DQN algorithm as well. We studied the MADRL system's performance under various conditions. We investigated data transmission on a cluster-based MAS. We proposed a novel immediate reward, including a new version of distance-based reward. We used three types of adversarial attacks to check the data transmission robustness of the MADRL system. We introduced two approaches to defend against malicious attacks and mitigate the destructive effects of adversarial attacks. The results of various scenarios were demonstrated and compared with each other numerically.

Chapter 7

Data Transmission Robustness to Cyber-attacks on Heterogeneous Multi-agent Deep Reinforcement Learning Systems

7.1 Introduction

This chapter investigates the data transmission robustness between agents of a cluster-based, heterogeneous, MADRL system under gradient-based adversarial attacks. We propose an algorithm using a DQN approach and a proportional feedback controller to defend against the FGSM attack and improve the DQN agent performance. The feedback control system is an auxiliary tool that helps the DQN algorithm reduce system deficiencies. In accordance with the achieved results and under FGSM adversarial attack, the robustness of the developed system is evaluated in three different ways termed *robust*, *semi-robust*, and *non-robust* based on average reward and DQN loss. The data transfer is carried out between agents of a MADRL system in a timely and time-delayed manners, for both leaderless and leader-follower scenarios. Simulation results are included to verify the presented results. The proposed algorithm in this chapter can be employed for data transmission between agents of a MARL system or a MADRL system to reach a consensus related to the Chapters 4 and 5.

7.2 Background

The graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ models leaderless and leader-follower MAS, where $\mathscr{V}$ is the set of all agents, and $\mathscr{E} \subseteq \{(i,j)|i \in \mathscr{V}, j \in \mathscr{V}\}$ represents all communication links between agents. A pair of agents i and j communicate if and only if $(i,j) \in \mathscr{E}$ [207]. In this chapter, all agents in a leaderless MAS scenario communicate with their cluster-mates and other cluster members. A MAS with leader-follower configurations allows only the preassigned leader within each cluster to communicate with its team-mates and other leaders within other clusters [2], [5], [218].

7.2.1 DQN Algorithm

The DQN algorithm observation set to transfer data between agents of MADRL system (leaderless or leader-follower) is $o_t = \{\text{busy, idle, on-time, time-delay, collided}\}$ [2], [5], [163]. Data channels are first checked if they are idle or busy. Data can be successfully transferred on-time, with time-delay, or collided if the channel is idle.

When the channel is busy at the time step t, the DQN agent's action at time step $t+1$ is $a_{t+1} = 0$. If $R_c \in \{0, 1, 2, \dots, R_{c\max}\}$ is the length of packets, when the channel is idle at time step t, the DQN agent's action at time step $t+1$ is $a_{t+1} = R_c$. This type of action selection is uniform selection with probability ε. Moreover, when the channel is idle at time step t, the DQN agent's action at time step $t+1$ is

$$a_{t+1} = \underset{a_t \in \{0,1,2,\dots,R_{c\max}\}}{\arg\max} \sum_{i=1}^{N} Q^i\left(s_{t+1}, a_t; \theta^-\right), \tag{7.1}$$

where θ^- defines the target Q-value weight. This type of action selection is non-uniform selection with probability $1-\varepsilon$ [163].

The DQN target Q-value for a state-action pair (s_t, a_t) is usually defined by

$$\mathrm{Tar}_Q = R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right), \tag{7.2}$$

where $\gamma \in (0,1)$ is the discount factor, and $R(s_t, a_t, s_{t+1})$ is the cumulative reward function. Since in [2], [5], and [163] $R(s_t, a_t, s_{t+1}) = \frac{r_{t+1}}{T_d} \cdot \frac{1-\gamma^{T_d}}{1-\gamma}$, the DQN target Q-value of a state-action pair (s_t, a_t) is given by

$$\mathrm{Tar}_Q = \frac{r_{t+1}}{T_d} \cdot \frac{1-\gamma^{T_d}}{1-\gamma} + \gamma^{T_d} \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta^-\right), \tag{7.3}$$

where $Q\left(s_{t+1}, a_{t+1}; \theta^-\right)$ is the target network, r_{t+1} is the immediate reward function at time $t+1$, and $T_d \in \{1, 2, \ldots, R_{c\max}\}$ is the action time duration for on-time observation. By replacing T_d with $T_d' \in \{R_{c\max}+1, R_{c\max}+2, \ldots\}$ the action time duration for time-delay observation is obtained [2], [5]. Moreover, the parameter θ is updated as

$$\theta_{t+1} \leftarrow \theta_t + \alpha \left(\mathrm{Tar}_Q - Q(s_t, a_t; \theta)\right) \nabla_\theta Q(s_t, a_t; \theta), \tag{7.4}$$

where α is the learning rate, and $Q(s_t, a_t; \theta)$ is the Q-value function.

For calculating the average loss, the DQN loss function is given by

$$\ell(\theta, s, a) = \frac{1}{B_e \cdot N} \sum_{i=1}^{N} \sum_{\substack{e_t \\ T_d \in \{1,2,\ldots,R_{c\max}\}}} \left(\frac{r_{t+1}^i}{T_d} \cdot \frac{1-\gamma^{T_d}}{1-\gamma} + \gamma^{T_d} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) - Q^i\left(s_t, a_t; \theta\right) \right)^2, \tag{7.5}$$

for experience $e_t = (s_t, a_t, T_d, r_{t+1}, s_{t+1})$ at time t. The average loss computations based on experience e_t are performed from $m = 1$ to B_e as the experience replay mini-batch size.

7.2.2 Linear Feedback Control System

The proportional feedback controller is given by

$$u(t) = K_p e(t), \tag{7.6}$$

where the constant K_p is proportional gain, and $e(t)$ is the tracking error.

7.2.3 FGSM Adversarial Attack

By targeting the system's input, the FGSM attack introduced by [169] causes the average loss to increase. The FGSM attack signal η is calculated by

$$\eta = \varepsilon \cdot \text{sign}\left(\nabla_s \ell(\theta, s, a)\right), \tag{7.7}$$

$$s_{adv} = s + \eta, \tag{7.8}$$

where ε is the attack magnitude, sign(.) is the sign function, ∇_s is the gradient of the loss function based on the model state s, ℓ is the DQN algorithm's loss function, θ is the model parameters, and s_{adv} is the adversarial input.

7.3 Methodology

This section introduces the combination of the DQN algorithm process and a linear feedback control system by considering different types of immediate reward functions in terms of on-time and time-delay data transmission. Then, the data transmission robustness of the MADRL system is presented.

7.3.1 DQN Algorithm and a Linear Feedback Control System

A combination of the DRL algorithm and a linear feedback control system is considered in the DQN agent learning process to defend against the FGSM adversarial attack in the MADRL system. According to the structure and learning process, a proportional controller is designed to help the DQN agent to eliminate the destructive effects of the FGSM attack. As demonstrated in Figure 7.1, the training procedure of the entire DRL algorithm is considered as the process part of the feedback control system. The designed proportional controller output $u(t)$ is specified by (7.6). The tracking error value $e(t)$ is given by

$$e(t) = \hat{r}^i_{t+1} - r^i_{t+1}, \tag{7.9}$$

where $\hat{r}^i_{t+1}$ and r^i_{t+1} are the optimal (estimated) and environmental received rewards at time step $t+1$, respectively. The feedback control system output of (7.6) is given to the DQN agent. During policy training and at time step t, the DNN of the DQN agent considers the controller output, along with its input, to select the most appropriate action. Depending on the different states that may exist in the system, the immediate reward for each agent is calculated using one of the methods presented in [2], [5], and [163] for on-time and time-delay data transmission configurations. The optimal reward $\hat{r}^i_{t+1}$ at time step $t+1$ is calculated using the proposed technique in Appendix B of [163]. Based on the various immediate rewards, the following control outputs are calculated. The proportional controller output when the data packet transferred by each agent in the network successfully and on-time is given by

$$u(t) = K_p(\hat{r}^i_{t+1} - (R^i_c - H^i_p)). \tag{7.10}$$

When the data packet is transferred by each agent in the network successfully and by time-delay then,

$$u(t) = K_p(\hat{r}^i_{t+1} - (R^i_c - \kappa \cdot H^i_p)). \tag{7.11}$$

As a reminder, R_c is the packet length, H_p is the packet's header duration, and $\kappa \in (1, \infty)$ is a real

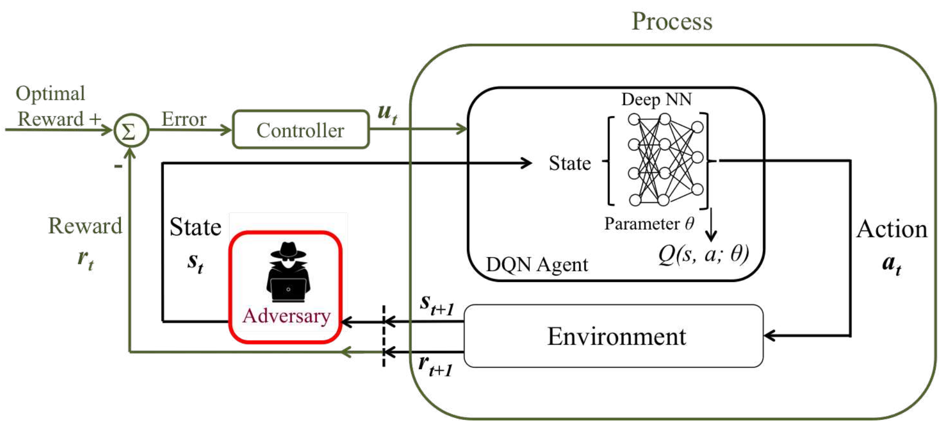


Figure 7.1: A feedback control system is used on the DQN learning process to assist the DRL system against the FGSM adversarial attack.

constant number for time-delayed data transmission. For on-time data transmission $\kappa = 1$. Note that $i \in \{1, 2, \ldots, N\}$ determines a specific agent. The proposed Algorithm 7.1 shows the DQN algorithm and feedback control system combination.

Theorem 7.1 *Assume that the MAS with N agents is modeled by a graph $\mathscr{G}$, and the learning process is carried out by DQN algorithm with the proportional feedback controller $u(t) = K_p e(t)$. With on-time observation for data transmission, the average approximated cumulative team discounted reward of a state-action pair (s_t, a_t) satisfies the following*

$$\frac{1}{N}\sum_{i=1}^{N} \underset{u(t)=0}{Q^i(s_t, a_t; \theta)} \leq \frac{1}{N}\sum_{i=1}^{N} \underset{u(t)=K_p e(t)}{Q^i(s_t, a_t; \theta)}. \tag{7.12}$$

Proof 7.1 *For an on-time data transmission, we consider the action time duration $T_d \in \{1, 2, \ldots, R_{c\max}\}$. Since in this work $\hat{r}^i_{t+1}$ is a real constant number at each time step, and $\hat{r}^i_{t+1} \gg r^i_{t+1}$, then $K_p e(t) \geq 0$ ($K_p > 0$, and $e(t) \geq 0$). According to [2], [5], [163] and using (7.6) the following is given*

$$\sum_{i=1}^{N} \frac{R^i_c - H^i_p}{T^i_d} \leq \sum_{i=1}^{N} \frac{R^i_c - H^i_p + K_p e(t)}{T^i_d}. \tag{7.13}$$

Algorithm 7.1 DQN Algorithm as the Feedback Control System Process to Defend Against FGSM Cyber-attack

```
    Input: $t$, $T_{max}$, $\theta$, $\varepsilon$, $s$, $\kappa$, $K_p$, $K_i$, $K_d$, setpoint, $o_t$, $a_0$, $r_0$, $\hat{r}_{t+1}$
    Output: $s_{adv}$, $a_{t+1}$, $r_{t+1}$, $u(t)$
  $o_t$ ={busy, idle, on-time, time-delay, collided}
  $a_0 = 0$
  $r_0 = 0$
  $\hat{r}_{t+1} \in \mathbb{R}$
  setpoint $\longleftarrow \hat{r}_{t+1}$
  for $t = 0, 1, 2, \ldots, T_{max}$ do
      if attack = FGSM then
          $\eta = \varepsilon \cdot \text{sign}\left(\nabla_s \ell(\theta, s, a)\right)$
          $s_{adv} = s + \eta$
      end if
      while attack = True do
          if observation is busy then
              $a_{t+1} = 0$
          else if observation is idle then
              $a_{t+1} \neq 0$
              $\begin{cases} a_{t+1} = R_c \in \{0, 1, 2, \ldots, R_{c\max}\} \\ a_{t+1} = \underset{a_t \in \{0,1,2,\ldots,R_{c\max}\}}{\arg\max} \sum_{i=1}^{N} Q^i\left(s_{t+1}, a_t; \theta^-\right), \end{cases}$
              if idle observation is on-time then
                  $r_{t+1} = R_c^i - H_p^i$

              else if idle observation is time-delay then
                  $r_{t+1} = R_c^i - (\kappa \cdot H_p^i)$
              end if

              $e(t)$ =setpoint$-r_{t+1}$
              $u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$
              $u(t) \longleftarrow$ update $(u(t))$
              while setpoint $> 0$ and $e(t) \geq 0$ do
                  $r_{t+1} \longleftarrow r_{t+1} + u(t)$
              end while
          end if
      end while
  end for
```

By assuming a specific value of $\gamma \in (0,1)$, the following inequality is provided

$$\sum_{i=1}^{N} \frac{R_c^i - H_p^i}{T_d^i} \cdot \frac{1-\gamma^{T_d^i}}{1-\gamma} \leq \sum_{i=1}^{N} \frac{R_c^i - H_p^i + K_p e(t)}{T_d^i} \cdot \frac{1-\gamma^{T_d^i}}{1-\gamma}. \tag{7.14}$$

Furthermore, utilizing the discount factor $\gamma \in (0,1)$ and action time duration T_d, the following is valid

$$0 < \underset{T_d \in \{1,2,\ldots,R_{c\max}\}}{\gamma^{T_d}} < 1. \tag{7.15}$$

Consequently, for N agents, we have

$$\underset{T_d^i \in \{1,2,\ldots,R_{c\max}\}}{\sum_{i=1}^{N} \gamma^{T_d^i}} > 0. \tag{7.16}$$

If setpoint $\in (0,\infty)$, then the target network is considered by

$$Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) > 0. \tag{7.17}$$

Based on (7.16), and by considering the maximum of target network, we obtain

$$\underset{T_d^i \in \{1,2,\ldots,R_{c\max}\}}{\sum_{i=1}^{N} \gamma^{T_d^i} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right)} > 0. \tag{7.18}$$

Using (7.14) and (7.18), the following inequalities are obtained

$$\begin{aligned} &\sum_{i=1}^{N} \frac{R_c^i - H_p^i}{T_d^i} \cdot \frac{1-\gamma^{T_d^i}}{1-\gamma} + \gamma^{T_d^i} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right) \leq \\ &\sum_{i=1}^{N} \frac{R_c^i - H_p^i + K_p e(t)}{T_d^i} \cdot \frac{1-\gamma^{T_d^i}}{1-\gamma} + \gamma^{T_d^i} \max_{a_{t+1}} Q^i\left(s_{t+1}, a_{t+1}; \theta^-\right), \end{aligned} \tag{7.19}$$

$$\underset{u(t)=0}{\sum_{i=1}^{N} \mathrm{Tar}_{Q^i}} \leq \underset{u(t)=K_p e(t)}{\sum_{i=1}^{N} \mathrm{Tar}_{Q^i}}. \tag{7.20}$$

The difference between the target Q-value and predicted Q-value should converge to zero to obtain the minimum amount of training loss $\ell(\theta,s,a)$ [2], [5]. Accordingly, the following equation applies to i^{th} agent,

$$\lim_{t\to t_0} Q^i(s_t,a_t;\theta) = \mathrm{Tar}_{Q^i}\,. \tag{7.21}$$

Substituting (7.21) in (7.20) yields

$$\sum_{i=1}^{N} \underset{u(t)=0}{\lim_{t\to t_0} Q^i(s_t,a_t;\theta)} \le \sum_{i=1}^{N} \underset{u(t)=K_p e(t)}{\lim_{t\to t_0} Q^i(s_t,a_t;\theta)}. \tag{7.22}$$

Regarding the monotone convergence condition, the following is valid

$$\sum_{i=1}^{N} \lim_{t\to t_0} Q^i(s_t,a_t;\theta) = \lim_{t\to t_0} \sum_{i=1}^{N} Q^i(s_t,a_t;\theta). \tag{7.23}$$

Using (7.23), the inequality (7.22) is altered as following

$$\underset{u(t)=0}{\lim_{t\to t_0} \sum_{i=1}^{N} Q^i(s_t,a_t;\theta)} \le \underset{u(t)=K_p e(t)}{\lim_{t\to t_0} \sum_{i=1}^{N} Q^i(s_t,a_t;\theta)}, \tag{7.24}$$

$$\underset{u(t)=0}{\sum_{i=1}^{N} Q^i(s_{t_0},a_{t_0};\theta)} \le \underset{u(t)=K_p e(t)}{\sum_{i=1}^{N} Q^i(s_{t_0},a_{t_0};\theta)}. \tag{7.25}$$

By redistributing each side of the inequality (7.25) to time t and taking an average from both sides of the relation, the following is provided

$$\underset{u(t)=0}{\frac{1}{N}\sum_{i=1}^{N} Q^i(s_t,a_t;\theta)} \le \underset{u(t)=K_p e(t)}{\frac{1}{N}\sum_{i=1}^{N} Q^i(s_t,a_t;\theta)}. \tag{7.26}$$

Therefore, the inequality (7.12) is proven.

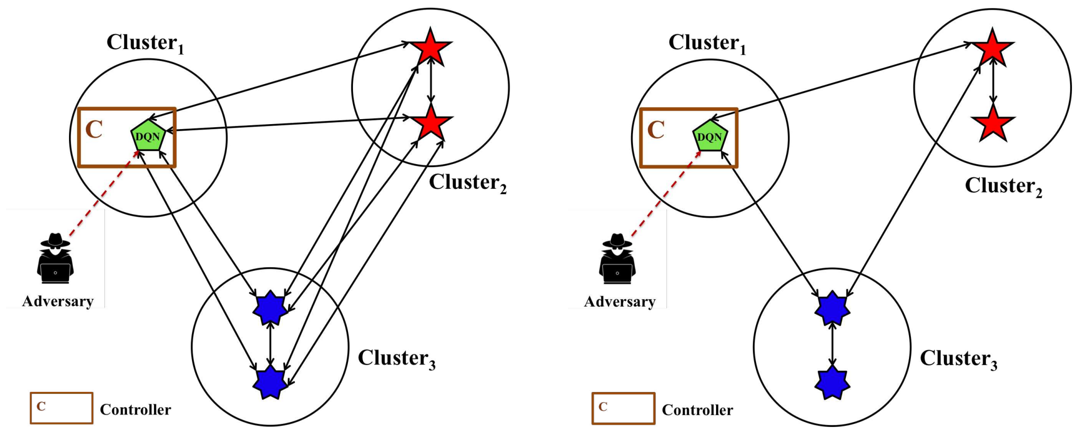
Remark 7.1 *Since $T'_d \in \{R_{c\max}+1, R_{c\max}+2,\ldots\}$ is considered as the time-delayed action time duration in [2], and [5], then the proof of Theorem 7.1 can be expanded to show that using a pro-*

portional controller for time-delay observation causes to produce a greater average approximated cumulative team discounted reward rather than the absence of the feedback controller.

7.3.2 Data Transmission Robustness Evaluation

In the absence of an attack, the DQN algorithm aims to reduce the average loss in a given time step and enhance the average reward. The purpose of the FGSM attack is to increase the average loss and thus reduce the average reward. Hence, under FGSM attack, the robustness of our developed system is evaluated in three ways based on the DQN algorithm's aim and FGSM attack's purpose: (i) *robust:* the average loss is not increased and the average reward is not decreased compared to the non-attack mode; (ii) *semi-robust:* the average loss is increased, but the average reward is not decreased compared to the non-attack mode and vice versa; (iii) *non-robust:* the average loss is increased and the average reward is decreased compared to the non-attack mode.

7.4 Results and Discussion



(a) Graph $\mathscr{G}$ as a leaderless MAS. (b) Graph $\mathscr{G}$ as a leader-follower MAS.

Figure 7.2: Leaderless and leader-follower heterogeneous MAS, including DQN, ALOHA, and TDMA structures, affected by a cyber-attack.

This section examines the data transfer robustness between agents in the MADRL system under the FGSM malicious attack with respect to average team reward and DQN loss. The investigation is considered in various conditions, e.g., on-time and time-delay data transmission. The simulation

Table 7.1: Utilized K_p gain after manual tuning under FGSM cyber-attack.

Proportional Controller Gain	
Various Graphs \ **Proportional Gain**	K_p
Leaderless MAS	0.0050
Leaderless MAS with time-delay	0.00575
Leader-follower MAS	0.0059
Leader-follower MAS with time-delay	0.00557

examples and data analysis are shown later.

For undirected leaderless and leader-follower scenarios (Figure 7.2), the DQN agent uses a feed-forward NN architecture for training, consisting of 36 layers with Adam optimization and MSE loss functions. All 36 layers have a rectified linear unit (ReLU) as their activation function. As an additional detail, DQN learning rate is $\alpha = 0.01$, discount factor is $\gamma = 0.999$, the experience-replay mini-batch size is $B_e = 64$, and constant positive real number for calculating immediate reward r^i_{t+1} if the data packet is transferred in the network by delay is $\kappa = 4$. A threshold of 11 mini-slots determines on-time or time-delay data transmission. The $\varepsilon = 0.6$ is assumed as the attack magnitude to calculate the attack signal. Simulation scenarios are carried out during 10000 time steps such that the FGSM malicious attack occurs on the DQN agent after starting the training process. The DQN algorithm setpoint that is considered as the estimated reward is 0.255.

Table 7.1 shows the different values of proportional gain K_p for various scenarios obtained by manual tuning. In order to ensure the validity of the outcomes, all the results are shown after five training cycles. Since the agents' structures of each cluster in the cluster-based MADRL system are different, the first, second, and third clusters of agents' architecture are the DQN algorithm, the ALOHA protocol, and the TDMA protocol, respectively.

As part of our implementation, we have used, modified, and extended parts of the code provided in [213]. In addition, the algorithm is executed on a system equipped with a 3.60 GHz Intel Core $i7-7700$ processor, 16GB of RAM, a $64-$bit operating system, and an $x64-$based processor.

7.4.1 DQN Algorithm's Robustness

The destructive effect of the FGSM adversarial attack on the MADRL system is visible in Table 7.2 and Table 7.3 in such a way that in all scenarios, after attack, the average loss of the DQN algorithm has increased.

The average loss of leaderless MAS has increased by 69.93% (from 21715.09 to 36901.97), and its average team reward has decreased by 16.94% (from 0.0608 to 0.0505). Therefore, without a proportional controller, the DQN algorithm of leaderless MAS is not robust to FGSM attack for data transmission and is located in the *non-robust* category. The average loss of leaderless MAS with time-delay has enhanced by +23.85% (from 15905.47 to 19698.95), and its average team reward has increased by +1.65% (from 0.0604 to 0.0614). Hence, without a proportional controller, the DQN algorithm of leaderless MAS with time-delay is partly robust to FGSM attack for data transmission and is placed in the *semi-robust* category.

Moreover, the average loss of leader-follower MAS has increased by 25.50% (from 17161.44 to 21537.88), and its average team reward has decreased by 1.83% (from 0.0598 to 0.0587). Therefore, without a proportional controller, the DQN algorithm of leader-follower MAS is not robust to the FGSM attack for data transmission and is located in the *non-robust* category. The average loss of leader-follower MAS with time-delay has increased by 155.07% (from 16297.44 to 41570.27), and its average team reward has enhanced by 9.24% (from 0.0530 to 0.0579). Hence, without a proportional controller, the DQN algorithm of leader-follower MAS with time-delay is partly robust to the FGSM attack for data transmission and is positioned in the *semi-robust* category.

Under the FGSM attack, the leaderless with time-delay and leader-follower with time-delay scenarios fall into the *semi-robust* category. The reason is that the delay in data transmission gives enough time to the MADRL system to identify the cyber-attack and ignore its destructive effects as much as possible.

Table 7.2: Comparison of DQN loss and average reward of $N = 5$ agents in a cluster-based MADRL system with $P = 3$ clusters before FGSM cyber-attack during 10000 time steps.

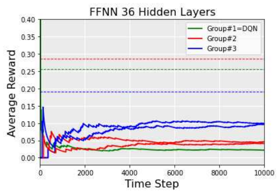
DQN Loss and Average Reward Before FGSM Attack							
Various Graphs / **Agents**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team Reward**	**DQN Loss**
Leaderless MAS	0.0218	0.0458	0.0411	0.0962	0.0993	0.0608	21715.09
Leaderless MAS with time-delay	0.0172	0.0380	0.0427	0.0950	0.1092	0.0604	15905.47
Leader-follower MAS	0.0172	0.0463	0.0477	0.0974	0.0904	0.0598	17161.44
Leader-follower MAS with time-delay	0.0087	0.0427	0.0522	0.0570	0.1045	0.0530	16297.44

Table 7.3: Comparison of DQN loss and average reward of $N = 5$ agents in a cluster-based MADRL system with $P = 3$ clusters under FGSM cyber-attack during 10000 time steps.

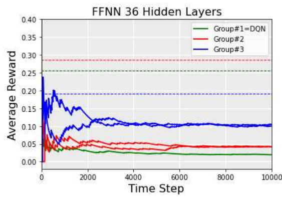
DQN Loss and Average Reward Under FGSM Attack Using $\varepsilon = 0.6$							
Various Graphs / **Agents**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team Reward**	**DQN Loss**
Leaderless MAS	0.0245	0.0237	0.0522	0.0807	0.0712	0.0505	36901.97
Leaderless MAS with time-delay	0.0127	0.0427	0.0617	0.0760	0.1140	0.0614	19698.95
Leader-follower MAS	0.0132	0.0475	0.0380	0.0902	0.1045	0.0587	21537.88
Leader-follower MAS with time-delay	0.0140	0.0570	0.0142	0.1092	0.0950	0.0579	41570.27

Table 7.4: Comparison of DQN loss and average reward of $N = 5$ agents in a cluster-based MADRL system with $P = 3$ clusters under FGSM cyber-attack during 10000 time steps using DRL and proportional controller.

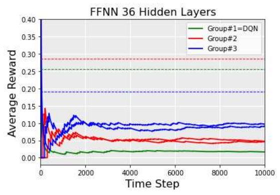
DQN Loss and Average Reward Under FGSM Attack Using $\varepsilon = 0.6$							
Various Graphs / **Agents**	**Agent 1**	**Agent 2**	**Agent 3**	**Agent 4**	**Agent 5**	**Team Reward**	**DQN Loss**
Leaderless MAS	0.2002	0.0570	0.0380	0.0997	0.1045	0.0999	19516.75
Leaderless MAS with time-delay	0.2778	0.0477	0.0486	0.0851	0.0973	0.1113	14676.98
Leader-follower MAS	0.244	0.0522	0.0332	0.1045	0.0997	0.1067	13984.18
Leader-follower MAS with time-delay	0.2717	0.0332	0.0665	0.076	0.0807	0.1056	25682.28



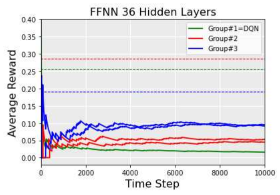
(a) Leaderless MAS.



(b) Leaderless MAS with time-delay.

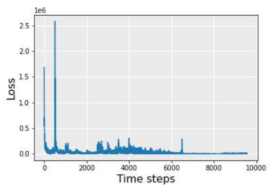


(c) Leader-follower MAS.

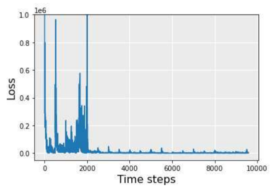


(d) Leader-follower MAS with time-delay.

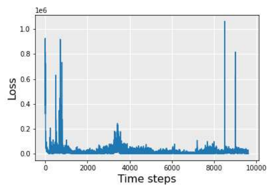
Figure 7.3: Reward convergence of $N = 5$ agents in a cluster-based MADRL system, including $P = 3$ various clusters, before FGSM cyber-attack during 10000 time steps.



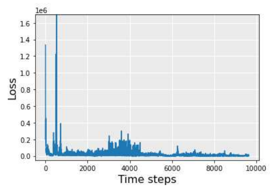
(a) Leaderless MAS.



(b) Leaderless MAS with time-delay.

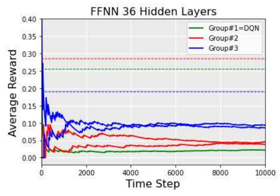


(c) Leader-follower MAS.

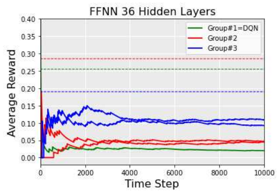


(d) Leader-follower MAS with time-delay.

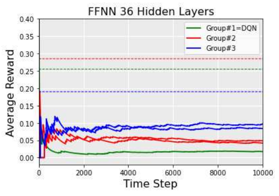
Figure 7.4: Loss convergence of the DQN algorithm in a cluster-based MADRL system, including $P = 3$ various clusters, before FGSM cyber-attack during 10000 time steps.



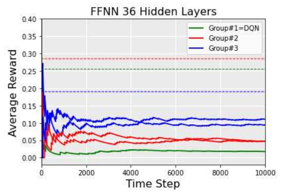
(a) Leaderless MAS.



(b) Leaderless MAS with time-delay.

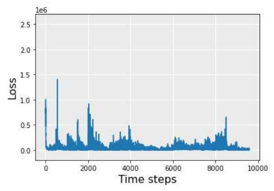


(c) Leader-follower MAS.

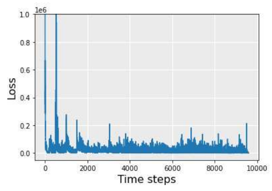


(d) Leader-follower MAS with time-delay.

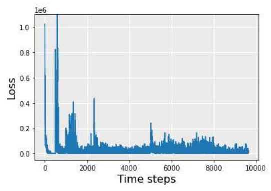
Figure 7.5: Reward convergence of $N = 5$ agents in a cluster-based MADRL system, including $P = 3$ various clusters, under FGSM cyber-attack during 10000 time steps with $\varepsilon = 0.6$.



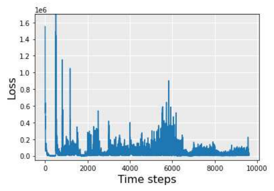
(a) Leaderless MAS.



(b) Leaderless MAS with time-delay.

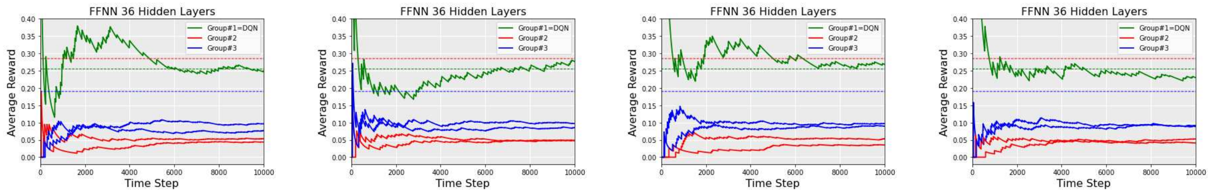


(c) Leader-follower MAS.



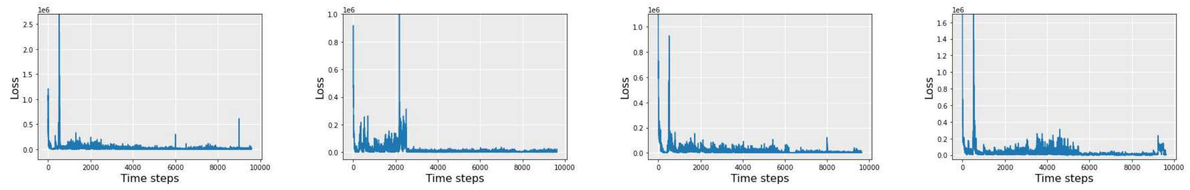
(d) Leader-follower MAS with time-delay.

Figure 7.6: Loss convergence of the DQN algorithm in a cluster-based MADRL system, including $P = 3$ various clusters, under FGSM cyber-attack during 10000 time steps with $\varepsilon = 0.6$.



(a) Leaderless MAS. (b) Leaderless MAS with time-delay. (c) Leader-follower MAS. (d) Leader-follower MAS with time-delay.

Figure 7.7: Reward convergence of $N = 5$ agents in a cluster-based MADRL system, including $P = 3$ various clusters, under FGSM cyber-attack during 10000 time steps with $\varepsilon = 0.6$, using DRL and proportional controller.



(a) Leaderless MAS. (b) Leaderless MAS with time-delay. (c) Leader-follower MAS. (d) Leader-follower MAS with time-delay.

Figure 7.8: Loss convergence of the DQN algorithm in a cluster-based MADRL system, including $P = 3$ various clusters, under FGSM cyber-attack during 10000 time steps with $\varepsilon = 0.6$, using DRL and proportional controller.

7.4.2 DQN Algorithm's Robustness: Proportional Controller

The positive effect of the proportional controller on the MADRL system under the FGSM adversarial attack is apparent in Table 7.3 and Table 7.4. By comparing the MAS results of average loss and average team reward in Tables 7.3 and 7.4, the outcomes below are achieved.

The average loss of leaderless MAS has reduced by 47.11% (from 36901.97 to 19516.75), and its average team reward has increased by 97.82% (from 0.0505 to 0.0999). Therefore, with a proportional controller, the DQN algorithm of leaderless MAS is robust to FGSM attack for data transmission rather than the absence of a proportional controller and is located in the *robust* category. Furthermore, the average loss of leaderless MAS with time-delay has diminished by 25.49% (from 19698.95 to 14676.98), and its average team reward has enhanced by 81.27% (from 0.0614 to 0.1113). Hence, with a proportional controller, the DQN algorithm of leaderless MAS with time-delay is robust to the FGSM attack for data transmission rather than the absence of a proportional controller and is placed in the *robust* category.

Moreover, the average loss of leader-follower MAS has decreased by 35.07% (from 21537.88 to 13984.18), and its average team reward has increased by 81.77% (from 0.0587 to 0.1067). Therefore, with a proportional controller, the DQN algorithm of leader-follower MAS is robust to the FGSM attack for data transmission rather than the absence of a proportional controller and is placed in the *robust* category. The average loss of leader-follower MAS with time-delay has reduced by 38.21% (from 41570.27 to 25682.28), and its average team reward has enhanced by 82.38% (from 0.0579 to 0.1056). Hence, with a proportional controller, the DQN algorithm of leader-follower MAS with time-delay is robust to the FGSM attack for data transmission rather than the absence of a proportional controller and is positioned in the *robust* category.

In our proposed method, combining the DQN algorithm with a proportional controller improves the team performance for all scenarios compared to when only the DQN algorithm is used. Adding a proportional controller diminishes the number of *non-robust* and *semi-robust* systems under FGSM adversarial attack in data transmission. The results of Tables 7.2-7.4 are shown in Figures 7.3-7.8. The large fluctuations in the amount of loss for various scenarios in Figure 7.6 rather than Figure 7.4 are due to FGSM adversarial attack. As can be seen from Figure 7.8, the fluctuation in the amount of loss after using the proportional feedback controller is reduced compared to Figure 7.6.

Compared to the other research developments using the pure DQN algorithm [2], in our experiment, the proportional feedback control system based on the DQN algorithm can improve the MAS performance and overcome the system disturbances caused by cyber-attacks, especially FGSM adversarial attacks, in a lower period of time.

It is worth mentioning that Python Programming Language is used to develop and simulate the algorithm of this chapter.

7.5 Conclusions

We studied the data transmission robustness to cyber-attacks on an undirected leaderless or leader-follower, heterogeneous, cluster-based MADRL system using the DQN algorithm by considering on-time and time-delay observations. After designing a proportional controller and adding it to the DQN learning process, we concentrated on the MADRL system's performance and data transmission robustness in terms of average team reward and DQN loss. The simulation results of various scenarios were presented and analysed.

Chapter 8

Summary and Future Works

8.1 Summary

In this dissertation, we considered the average position consensus of cluster-based, heterogeneous MAS using the non-learning methods. We presented the discussed average position consensus of agents' clusters in two scenarios for 2-D and 3-D spaces. The average position consensus was performed when each cluster's agents had the same goal; however, its goal differed from the other groups. Later, the average position consensus was done when the aim of all clusters' agents was unique. The successful position consensus for the proposed cluster-based, heterogeneous MAS in 2-D and 3-D spaces was shown by numerical and simulation results.

We studied the consensus problem of a leaderless, homogeneous MARL system using the actor-critic algorithms in the absence and presence of malicious agents. Each agent's principal goal was to reach the position consensus with the maximum cumulative reward. We presented the immediate reward function based on Manhattan distance. Then, we proposed three other immediate reward functions based on various distance metrics to improve the MARL system's performance. We combined different immediate reward functions and used each of them based on the maximum returned value during each episode to enhance agents' cumulative reward in the presence of malicious agents within the MARL system. Finally, we compared different immediate reward functions within the MARL system. We found that the type of immediate reward function plays a significant role in the efficiency of each agent in the network in reaching the consensus and obtaining further cumulative team rewards. The claims were proved theoretically, and the simulation confirmed theoretical findings.

Moreover, in this dissertation, we studied the control of a leaderless, homogeneous MARL system using actor-critic algorithms in the presence of a malicious agent. Using the gradient of the loss function with respect to the activation function, we proved that when the MSE loss function was combined with the sigmoid activation function in the malicious agent's critic and reward NN (last layer), the loss of `Agent#1` (malicious agent) was increased. Then, the cumulative reward decreased dramatically compared to when the linear activation function was used in the last layer of the adverse agent's critic and reward NN. The simulation results confirmed the theoretical analysis.

We studied the on-time and time-delayed data transmission of a leaderless, heterogeneous MADRL system using the DQN algorithm. We also investigated the on-time and time-delayed data transmission of a leader-follower, heterogeneous MADRL system using the DQN algorithm. We studied the MADRL system's performance under various conditions. We did the data transmission investigation on a cluster-based MAS framework. We proposed a novel immediate reward, including a new version of the distance-based reward. We used three types of adversarial attacks to check the data transmission robustness of the MADRL system. We introduced two approaches for defending against malicious attacks and mitigating the destructive effects of adversarial attacks. The results of various scenarios were demonstrated and compared with each other numerically.

We investigated the data transmission robustness to cyber-attacks on an undirected leaderless or leader-follower, heterogeneous, cluster-based MADRL system using the DQN algorithm by considering on-time and time-delay observations. After designing a proportional controller and adding it to the DQN algorithm learning process, we concentrated on the MADRL system's performance and data transmission robustness regarding average team reward and DQN loss. The simulation results of various scenarios were demonstrated and compared with each other.

8.2 Future Work

The possible future works related to this dissertation are:

- Proposing various adversarial attack detection methods:

  RL- and DRL-based adversarial attack detection algorithms can be studied for the various types of malicious attacks that target the introduced processes in this dissertation. Moreover, the computer-vision object detection approaches could be extended and used for the gradient-based attacks considered in this thesis or other possible attacks.

- Extending the proposed algorithms in the presence of obstacles:

  Various RL- and DRL-based object detection algorithms can be proposed to avoid the possible static or dynamic obstacles in the way of agents' movement (if the agents are mobile) in a MAS. Collision avoidance can be considered in two cases as below:

  Internal Obstacle: In the first case, each agent considers the other neighbouring agents as obstacles and does not get closer to them. Therefore, a certain distance between neighbouring agents must be considered as below:

$$\begin{cases} d(i,j) \leq d_{in}, \\ d(k,l) \leq d_{out}, \end{cases}$$

  where $d(i,j)$ is the distance between the i^{th} and j^{th} neighbouring agents in a cluster, and $d(k,l)$ is the distance between the k^{th} and l^{th} neighbouring agents of two neighbouring clusters. Furthermore, d_{in} and d_{out} are constant defined values.

  External Obstacle: In the second case, static or dynamic objects outside the MAS are considered as obstacles. Therefore, agents during movement must avoid colliding with static or dynamic obstacles. In this regard, it may be possible to consider each agent in the center of a circle with a radius r. If agents can detect a foreign object at a distance r or closer, they must react and avoid colliding with that obstacle.

Bibliography

[1] H. Ji, O. Alfarraj, and A. Tolba, “Artificial intelligence-empowered edge of vehicles: architecture, enabling technologies, and applications,” *IEEE Access*, vol. 8, pp. 61020–61034, 2020.

[2] N. Elhami Fard and R. R. Selmic, “Adversarial attacks on heterogeneous multi-agent deep reinforcement learning system with time-delayed data transmission,” *Journal of Sensor and Actuator Networks*, vol. 11, no. 3, p. 45, 2022.

[3] N. Elhami Fard and R. R. Selmic, “Consensus of multi-agent reinforcement learning systems: The effect of immediate rewards,” *Journal of Robotics and Control (JRC)*, vol. 3, no. 2, pp. 115–127, 2022.

[4] N. Elhami Fard and R. R. Selmic, “Data transmission resilience to cyber-attacks on heterogeneous multi-agent deep reinforcement learning systems,” in *Proceedings of The 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2022.

[5] N. Elhami Fard and R. R. Selmic, “Time-delayed data transmission in heterogeneous multi-agent deep reinforcement learning system,” in *Proceedings of the 2022 30th Mediterranean Conference on Control and Automation (MED)*, pp. 636–642, 2022.

[6] V. Selvi and R. Umarani, “Comparative analysis of ant colony and particle swarm optimization techniques,” *International Journal of Computer Applications*, vol. 5, no. 4, pp. 1–6,

2010.

[7] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[8] B. D. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 48–63, 2008.

[9] M. Pachter and P. R. Chandler, "Challenges of autonomous control," *IEEE Control Systems Magazine*, vol. 18, no. 4, pp. 92–97, 1998.

[10] H. Xinhan and W. Min, "Multi-sensor data fusion structures in autonomous systems: a review," in *Proceedings of the 2003 IEEE International Symposium on Intelligent Control*, pp. 817–821, IEEE, 2003.

[11] Y. Tang, C. Zhao, J. Wang, C. Zhang, Q. Sun, W. X. Zheng, W. Du, F. Qian, and J. Kurths, "Perception and navigation in autonomous systems in the era of learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[12] W. Fink, J. Dohm, and M. A. Tarbell, "Multi-agent autonomous system," Jan. 24 2006. US Patent 6,990,406.

[13] W. Fink, J. Dohm, and M. A. Tarbell, "Multi-agent autonomous system and method," June 22 2010. US Patent 7,742,845.

[14] A. Rossiter, "The impact of robotics and autonomous systems (ras) across the conflict spectrum," *Small Wars & Insurgencies*, vol. 31, no. 4, pp. 691–700, 2020.

[15] J. J. Leonard, A. A. Bennett, C. M. Smith, H. Jacob, and S. Feder, "Autonomous underwater vehicle navigation," in *MIT Marine Robotics Laboratory Technical Memorandum*, Citeseer, 1998.

[16] P. A. Miller, J. A. Farrell, Y. Zhao, and V. Djapic, "Autonomous underwater vehicle navigation," *IEEE Journal of Oceanic Engineering*, vol. 35, no. 3, pp. 663–678, 2010.

[17] P. Encarnacao and A. Pascoal, "3d path following for autonomous underwater vehicle," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 3, pp. 2977–2982, IEEE, 2000.

[18] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi, "Seaglider: A long-range autonomous underwater vehicle for oceanographic research," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 424–436, 2001.

[19] A. Birk and S. Carpin, "Rescue robotics—a crucial milestone on the road to autonomous systems," *Advanced Robotics*, vol. 20, no. 5, pp. 595–605, 2006.

[20] F. J. R. Lera, C. F. Llamas, Á. M. Guerrero, and V. M. Olivera, "Cybersecurity of robotics and autonomous systems: Privacy and safety," *Robotics-Legal, Ethical and Socioeconomic Impacts*, 2017.

[21] J. Balsa-Comerón, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, C. Fernández-Llamas, and V. Matellán-Olivera, "Cybersecurity in autonomous systems: hardening ros using encrypted communications and semantic rules," in *Iberian Robotics Conference*, pp. 67–78, Springer, 2017.

[22] M. S. Rais, K. Zouaidia, and R. Boudour, "Enhanced decision making in multi-scenarios for autonomous vehicles using alternative bidirectional q network," *Neural Computing and Applications*, pp. 1–16, 2022.

[23] T. Linz, "Testing autonomous systems," in *The Future of Software Quality Assurance*, pp. 61–75, Springer, Cham, 2020.

[24] V. Kumar, "50 years of robotics [from the guest editors]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 8–8, 2010.

[25] M. Weber, "where to? a history of autonomous vehicles." https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/?key=where-to-a-history-of-autonomous-vehicles, May 2014.

[26] J. Von Neumann, N. Wiener, and S. J. Heims, *From mathematics to the technologies of life and death*. MIT press, 1981.

[27] D. P. Watson and D. H. Scheidt, "Autonomous systems," *Johns Hopkins APL Technical Digest*, vol. 26, no. 4, pp. 368–376, 2005.

[28] B. Pell, D. E. Bernard, S. Chien, E. Gat, N. Muscettola, P. P. Nayak, M. D. Wagner, and B. C. Williams, "Remote agent prototype for spacecraft autonomy," in *Space Sciencecraft Control and Tracking in the New Millennium*, vol. 2810, pp. 74–90, International Society for Optics and Photonics, 1996.

[29] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote agent: To boldly go where no ai system has gone before," *Artificial Intelligence*, vol. 103, no. 1-2, pp. 5–47, 1998.

[30] J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of Human-Robot Interaction*, vol. 3, no. 2, p. 74, 2014.

[31] "autonomy." https://www.merriam-webster.com/dictionary/autonomy, June 2020.

[32] "automatic." https://www.merriam-webster.com/dictionary/automatic, June 2020.

[33] "automation." https://www.merriam-webster.com/dictionary/automation, July 2020.

[34] H. Huang, "Autonomy levels for unmanned systems (alfus) framework–volume i: Terminology–version 2.0 (nist special publication 1011-i-2.0)," *Gaithersburg, MD, USA*, 2008.

[35] C. Ebert and M. Weyrich, "Validation of autonomous systems," *IEEE Software*, vol. 36, no. 5, pp. 15–23, 2019.

[36] W. Xu, "From automation to autonomy and autonomous vehicles: Challenges and opportunities for human-computer interaction," *Interactions*, vol. 28, no. 1, pp. 48–53, 2020.

[37] Caterpillar, "Automation and autonomy: What's the difference?." https://www.equipmentworld.com/partner-solutions-article/caterpillar/automation-autonomy-whats-the-difference/, May 2016.

[38] T. Pettinger, "Automation – benefits and costs." https://www.economicshelp.org/blog/25163/economics/automation/, Nov. 2019.

[39] "autonomous systems-advantages and disadvantages." https://sites.google.com/site/autonomoussystemsmw/autonomous-cars/advantages-and-disadvantages.

[40] J. Williamson, "What are the challenges for the development of autonomous systems?." https://www.themanufacturer.com/articles/challenges-development-autonomous-systems/, jun 2020.

[41] J. Hodicky and D. Prochazka, "Challenges in the implementation of autonomous systems into the battlefield," in *2017 International Conference on Military Technologies (ICMT)*, pp. 743–747, IEEE, 2017.

[42] J. Markoff, "Collision in the making between self-driving cars and how the world works," *New York Times*, 2012.

[43] R. C. Arkin, "Ethical robots in warfare," *IEEE Technology and Society Magazine*, vol. 28, no. 1, pp. 30–33, 2009.

[44] S. Bagchi, V. Aggarwal, S. Chaterji, F. Douglis, A. El Gamal, J. Han, B. Henz, H. Hoffmann, S. Jana, M. Kulkarni, *et al.*, "Vision paper: Grand challenges in resilience: Autonomous system resilience through design and runtime measures," *IEEE Annals of the History of Computing*, no. 01, pp. 1–1, 2020.

[45] Q. Ha, L. Yen, and C. Balaguer, "Robotic autonomous systems for earthmoving in military applications," *Automation in Construction*, vol. 107, p. 102934, 2019.

[46] "Emerging technologies." `https://en.wikipedia.org/wiki/Emerging-technologies`, Aug. 2020.

[47] M. Bayern, "The 5 emerging technologies worth investing in for 2020." `https://www.techrepublic.com/article/the-5-emerging-technologies-worth-investing-in-for-2020/`, aug 2019.

[48] R. Haddal and N. K. Hayden, "Autonomous systems artificial intelligence and safeguards.," tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM, 2018.

[49] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[50] B. Torossian, F. Bekkers, T. Sweijs, M. Roelen, A. Hristov, and S. Atalla, *The Military Applicability of Robotic and Autonomous Systems*. Hague Centre for Strategic Studies, 2020.

[51] M. Grolms, "Autonomous vehicles in logistics part 1: Opportunities and risks." `https://www.allthingssupplychain.com/autonomous-vehicles-in-logistics-part-1-opportunities-and-risks/`, jun 2020.

[52] O. David-West, “What do new autonomous technologies mean for global business?.” https://globalnetwork.io/perspectives/2016/09/what-do-new-autonomous-technologies-mean-global-business, sep 2016.

[53] H. He, J. Gray, A. Cangelosi, Q. Meng, T. McGinnity, and J. Mehnen, “The challenges and opportunities of artificial intelligence in implementing trustworthy robotics and autonomous systems,” in *3rd International Conference on Intelligent Robotic and Control Engineering*, 2020.

[54] J. A. Perez, F. Deligianni, D. Ravi, and G.-Z. Yang, “Artificial intelligence and robotics,” *arXiv preprint arXiv:1803.10813*, vol. 147, 2018.

[55] D. Rahmani and H. Kamberaj, “Implementation and usage of artificial intelligence powered chatbots in human resources management systems,” in *International Conference on Social and Applied SciencesAt: University of New York Tirana*, May 2021.

[56] S. Singh, “Cousins of artificial intelligence.” https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55, May 2018.

[57] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,” *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[59] I. H. Sarker, “Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective,” *SN Computer Science*, vol. 2, no. 3, pp. 1–16, 2021.

[60] X. Liu, H. Xu, W. Liao, and W. Yu, "Reinforcement learning for cyber-physical systems," in *2019 IEEE International Conference on Industrial Internet (ICII)*, pp. 318–327, May 2020.

[61] S. Mahdavifar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, 2019.

[62] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Transactions on Neural Networks and Learning Systems*, November 2019.

[63] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020.

[64] K. Sethi, R. Kumar, N. Prajapati, and P. Bera, "Deep reinforcement learning based intrusion detection system for cloud infrastructure," in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 1–6, IEEE, March 2020.

[65] P. M. Shakeel, S. Baskar, V. S. Dhulipala, S. Mishra, and M. M. Jaber, "Maintaining security and privacy in health care system using learning based deep-q-networks," *Journal of Medical Systems*, vol. 42, no. 10, pp. 1–10, 2018.

[66] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Defense strategies against adversarial jamming attacks via deep reinforcement learning," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, May 2020.

[67] L. Huang, M. Fu, H. Qu, S. Wang, and S. Hu, "A deep reinforcement learning-based method applied for solving multi-agent defense and attack problems," *Expert Systems with Applications*, vol. 176, p. 114896, 2021.

[68] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 2019.

[69] E. A. Lee, "The past, present and future of cyber-physical systems: A focus on models," *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[70] V. Fourkas, "What is cyberspace," *Spatial Development Research Unit, Department of Urban and Regional Planning and Development, Aristotle University of Thessalonica*, 2004.

[71] R. Alguliyev, Y. Imamverdiyev, and L. Sukhostat, "Cyber-physical systems and their security issues," *Computers in Industry*, vol. 100, pp. 212–223, 2018.

[72] E. L. Thorndike, "Animal intelligence: An experimental study of the associate processes in animals.," *American Psychologist*, vol. 53, no. 10, p. 1125, 1998.

[73] M. L. Minsky, *Theory of neural-analog reinforcement systems and its application to the brain model problem.* Princeton University., 1954.

[74] A. H. Klopf, *Brain function and adaptive systems: a heterostatic theory.* Air Force Cambridge Research Laboratories, Air Force Systems Command, United States, 1972.

[75] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT press Cambridge, 2018.

[76] K. D. Foote, "A brief history of deep learning." https://www.dataversity.net/brief-history-deep-learning/, February 2017.

[77] K. Kersandt, "Deep reinforcement learning as control method for autonomous uavs," Master's thesis, Universitat Politècnica de Catalunya, 2018.

[78] X. Huang, S. H. Hong, M. Yu, Y. Ding, and J. Jiang, "Demand response management for industrial facilities: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 82194–82205, 2019.

[79] A. Balakrishnan and J. Deshmukh, “Reinforcement learning for cyber-physical systems.” https://viterbi-web.usc.edu/~jdeshmuk/teaching/cs599-autocps-spring-2019/rl.pdf, March 2019.

[80] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, “Online cyber-attack detection in smart grid: A reinforcement learning approach,” *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5174–5185, 2018.

[81] Z. Wang, H. He, Z. Wan, and Y. Sun, “Coordinated topology attacks in smart grid using deep reinforcement learning,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1407–1415, 2020.

[82] X. Li, X. Liang, R. Lu, X. Shen, X. Lin, and H. Zhu, “Securing smart grid: cyber attacks, countermeasures, and challenges,” *IEEE Communications Magazine*, vol. 50, no. 8, pp. 38–45, 2012.

[83] G. Hug and J. A. Giampapa, “Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks,” *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, 2012.

[84] SDN, “What is sdn?.” https://www.ciena.com/insights/what-is/What-Is-SDN.html, June 2021.

[85] Y. Han, B. I. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague, “Reinforcement learning for autonomous defence in software-defined networking,” in *International Conference on Decision and Game Theory for Security*, pp. 145–165, Springer, September 2018.

[86] M. Feng and H. Xu, “Deep reinforecement learning based optimal defense for cyber-physical system in presence of unknown cyber-attack,” in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, February 2017.

[87] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 307–312, IEEE, December 2018.

[88] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, 2017.

[89] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Network*, vol. 32, no. 6, pp. 50–57, 2018.

[90] Y. Yamagata, S. Liu, T. Akazaki, Y. Duan, and J. Hao, "Falsification of cyber-physical systems using deep reinforcement learning," *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2823–2840, 2020.

[91] A. Gupta and Z. Yang, "Adversarial reinforcement learning for observer design in autonomous systems under cyber attacks," *arXiv preprint arXiv:1809.06784*, 2018.

[92] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 116–122, 2018.

[93] G. Han, L. Xiao, and H. V. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2087–2091, IEEE, June 2017.

[94] X. Liu, Y. Xu, L. Jia, Q. Wu, and A. Anpalagan, "Anti-jamming communications using spectrum waterfall: A deep reinforcement learning approach," *IEEE Communications Letters*, vol. 22, no. 5, pp. 998–1001, 2018.

[95] L. Xiao, Y. Li, G. Liu, Q. Li, and W. Zhuang, "Spoofing detection with reinforcement learning in wireless networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–5, February 2016.

[96] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10037–10047, 2016.

[97] X. Wan, G. Sheng, Y. Li, L. Xiao, and X. Du, "Reinforcement learning based mobile offloading for cloud-based malware detection," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, January 2018.

[98] Y. Li, J. Liu, Q. Li, and L. Xiao, "Mobile cloud offloading for malware detections with learning," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 197–201, IEEE, August 2015.

[99] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 35–47, 2017.

[100] M. Chatterjee and A.-S. Namin, "Detecting phishing websites through deep reinforcement learning," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 227–232, IEEE, July 2019.

[101] D. S. Hoadley and N. J. Lucas, "Artificial intelligence and national security," tech. rep., Congressional Research Service Washington, DC, 2018.

[102] B. Zhang, M. Anderljung, L. Kahn, N. Dreksler, M. C. Horowitz, and A. Dafoe, "Ethics and governance of artificial intelligence: Evidence from a survey of machine learning researchers," *Journal of Artificial Intelligence Research*, vol. 71, pp. 591–666, 2021.

[103] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[104] M. AlQuraishi, "Alphafold at casp13," *Bioinformatics*, vol. 35, no. 22, pp. 4862–4865, 2019.

[105] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[106] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362–370, 2018.

[107] 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.

[108] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS One*, vol. 12, no. 4, p. e0172395, 2017.

[109] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[110] R. Urena, F. Chiclana, G. Melancon, and E. Herrera-Viedma, "A social network based approach for consensus achievement in multiperson decision making," *Information Fusion*, vol. 47, pp. 72–87, 2019.

[111] R. Urena, G. Kou, Y. Dong, F. Chiclana, and E. Herrera-Viedma, "A review on trust prop-

agation and opinion dynamics in social networks and group decision making frameworks," *Information Sciences*, vol. 478, pp. 461–475, 2019.

[112] G. De Pasquale and M. E. Valcher, "Consensus for clusters of agents with cooperative and antagonistic relationships," *Automatica*, vol. 135, p. 110002, 2022.

[113] D. Shen, C. Zhang, and J.-X. Xu, "Distributed learning consensus control based on neural networks for heterogeneous nonlinear multiagent systems," *International Journal of Robust and Nonlinear Control*, vol. 29, no. 13, pp. 4328–4347, 2019.

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

[115] A. Wang, T. Dong, and X. Liao, "Distributed optimal consensus algorithms in multi-agent systems," *Neurocomputing*, vol. 339, pp. 26–35, 2019.

[116] B. Mu and Y. Shi, "Distributed lqr consensus control for heterogeneous multiagent systems: Theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 434–443, 2018.

[117] Z. Wang, J. Xu, X. Song, and H. Zhang, "Consensus problem in multi-agent systems under delayed information," *Neurocomputing*, vol. 316, pp. 277–283, 2018.

[118] Y. Wang, Y. Song, D. J. Hill, and M. Krstic, "Prescribed-time consensus and containment control of networked multiagent systems," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1138–1147, 2018.

[119] B. Wang, W. Chen, and B. Zhang, "Semi-global robust tracking consensus for multi-agent uncertain systems with input saturation via metamorphic low-gain feedback," *Automatica*, vol. 103, pp. 363–373, 2019.

[120] L. Liu, H. Sun, L. Ma, J. Zhang, and Y. Bo, "Quasi-consensus control for a class of time-varying stochastic nonlinear time-delay multiagent systems subject to deception attacks,"

IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 11, pp. 6863–6873, 2020.

[121] F. Shamsi, H. A. Talebi, and F. Abdollahi, "Output consensus control of multi-agent systems with nonlinear non-minimum phase dynamics," *International Journal of Control*, vol. 91, no. 4, pp. 785–796, 2018.

[122] J. Zhang, H. Zhang, and T. Feng, "Distributed optimal consensus control for nonlinear multiagent system with unknown dynamic," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3339–3348, 2017.

[123] L. Zha, J. Liu, and J. Cao, "Resilient event-triggered consensus control for nonlinear mutiagent systems with dos attacks," *Journal of the Franklin Institute*, vol. 356, no. 13, pp. 7071–7090, 2019.

[124] G. Cui, S. Xu, Q. Ma, Y. Li, and Z. Zhang, "Prescribed performance distributed consensus control for nonlinear multi-agent systems with unknown dead-zone input," *International Journal of Control*, vol. 91, no. 5, pp. 1053–1065, 2018.

[125] C. Gao, Z. Wang, X. He, and Q.-L. Han, "Consensus control of linear multiagent systems under actuator imperfection: When saturation meets fault," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[126] Z. Peng, J. Hu, K. Shi, R. Luo, R. Huang, B. K. Ghosh, and J. Huang, "A novel optimal bipartite consensus control scheme for unknown multi-agent systems via model-free reinforcement learning," *Applied Mathematics and Computation*, vol. 369, p. 124821, 2020.

[127] R. Moghadam and H. Modares, "Resilient adaptive optimal control of distributed multiagent systems using reinforcement learning," *IET Control Theory & Applications*, vol. 12, no. 16, pp. 2165–2174, 2018.

[128] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757v2*, 2018.

[129] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275v4*, 2020.

[130] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," *arXiv preprint arXiv:1810.02912v2*, 2019.

[131] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926v2*, 2017.

[132] M. Figura, K. C. Kosaraju, and V. Gupta, "Adversarial attacks in consensus-based multi-agent reinforcement learning," *arXiv preprint arXiv:2103.06967*, 2021.

[133] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *arXiv preprint arXiv:1807.04742v2*, 2018.

[134] Q.-K. Hu and Y.-P. Zhao, "Aero-engine acceleration control using deep reinforcement learning with phase-based reward function," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, p. 09544100211046225, 2021.

[135] Y. Lv, H. Zhang, Z. Wang, and H. Yan, "Distributed localization for multi-agent systems with random noise based on iterative learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–9, 2022.

[136] L. Liu, B. Li, and R. Guo, "Consensus control for networked manipulators with switched parameters and topologies," *IEEE Access*, vol. 9, pp. 9209–9217, 2021.

[137] Y. Cao and Y. Song, "Performance guaranteed consensus tracking control of nonlinear multiagent systems: A finite-time function-based approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 4, pp. 1536–1546, 2020.

[138] Y. Lv, H. Zhang, Z. Wang, and H. Yan, “Distributed localization for dynamic multiagent systems with randomly varying trajectory lengths,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 9, pp. 9298–9308, 2021.

[139] H. Yan, H. Zhang, F. Yang, X. Zhan, and C. Peng, “Event-triggered asynchronous guaranteed cost control for markov jump discrete-time neural networks with distributed delay and channel fading,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3588–3598, 2017.

[140] H. Zhang, J. Chen, Z. Wang, C. Fu, and S. Song, “Distributed event-triggered control for cooperative output regulation of multiagent systems with an online estimation algorithm,” *IEEE Transactions on Cybernetics*, 2020.

[141] H. Wang, “Flocking of networked uncertain euler–lagrange systems on directed graphs,” *Automatica*, vol. 49, no. 9, pp. 2774–2779, 2013.

[142] E. Nuno, “Consensus of euler-lagrange systems using only position measurements,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 489–498, 2016.

[143] L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques, “Leader–follower formation control of nonholonomic mobile robots with input constraints,” *Automatica*, vol. 44, no. 5, pp. 1343–1349, 2008.

[144] X. Lyu, Y. Xiao, B. Daley, and C. Amato, “Contrasting centralized and decentralized critics in multi-agent reinforcement learning,” *arXiv preprint arXiv:2102.04402*, 2021.

[145] Y. Jung, M. Kim, A. Masoumzadeh, and J. B. Joshi, “A survey of security issue in multi-agent systems,” *Artificial Intelligence Review*, vol. 37, no. 3, pp. 239–260, 2012.

[146] N. N. Bakhtadze, I. B. Yadykin, V. A. Lototsky, E. M. Maximov, and E. A. Sakrutina, “Multi-agent approach to design of multimodal intelligent immune system for smart grid,”

IFAC Proceedings Volumes, vol. 46, no. 9, pp. 1164–1169, 2013.

[147] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, “A survey of actor-critic reinforcement learning: Standard and natural policy gradients,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[148] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, “Multi-agent reinforcement learning: A review of challenges and applications,” *Applied Sciences*, vol. 11, no. 11, p. 4948, 2021.

[149] S. S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: an overview,” in *Proceedings of SAI Intelligent Systems Conference*, pp. 426–440, Springer, 2016.

[150] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” *arXiv preprint arXiv:1811.12560*, 2018.

[151] Z. Chen, S. Zhang, T. T. Doan, S. T. Maguluri, and J.-P. Clarke, “Performance of q-learning with linear function approximation: Stability and finite-time analysis,” *arXiv preprint arXiv:1905.11425*, 2019.

[152] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[153] Z. Gao, Y. Gao, Y. Hu, Z. Jiang, and J. Su, “Application of deep q-network in portfolio management,” in *Proceedings of the IEEE International Conference on Big Data Analytics (ICBDA)*, pp. 268–275, 2020.

[154] S. Carta, A. Ferreira, A. S. Podda, D. R. Recupero, and A. Sanna, “Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting,” *Expert Systems with Applications*, vol. 164, p. 113820, 2021.

[155] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, “Q-learning algorithms: A comprehensive classification and applications,” *IEEE Access*, vol. 7, pp. 133653–133667, 2019.

[156] J. Pan, X. Wang, Y. Cheng, and Q. Yu, “Multisource transfer double dqn based on actor learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2227–2238, 2018.

[157] C. Zhao, X. Liu, S. Zhong, K. Shi, D. Liao, and Q. Zhong, “Secure consensus of multi-agent systems with redundant signal and communication interference via distributed dynamic event-triggered control,” *ISA Transactions*, vol. 112, pp. 89–98, 2021.

[158] S. Zheng, P. Shi, R. K. Agarwal, and C. P. Lim, “Periodic event-triggered output regulation for linear multi-agent systems,” *Automatica*, vol. 122, p. 109223, 2020.

[159] S. Yuan, C. Yu, and J. Sun, “Adaptive event-triggered consensus control of linear multi-agent systems with cyber attacks,” *Neurocomputing*, vol. 442, pp. 1–9, 2021.

[160] M. Chen, H. Yan, H. Zhang, M. Chi, and Z. Li, “Dynamic event-triggered asynchronous control for nonlinear multi-agent systems based on ts fuzzy models,” *IEEE Transactions on Fuzzy Systems*, 2020.

[161] M. Rehan, M. Tufail, and S. Ahmed, “Leaderless consensus control of nonlinear multi-agent systems under directed topologies subject to input saturation using adaptive event-triggered mechanism,” *Journal of the Franklin Institute*, 2021.

[162] B. Zhou, Y. Yang, L. Li, and R. Hao, “Leaderless and leader-following consensus of heterogeneous second-order multi-agent systems on time scales: An asynchronous impulsive approach,” *International Journal of Control*, pp. 1–11, 2021.

[163] Y. Yu, S. C. Liew, and T. Wang, “Non-uniform time-step deep q-network for carrier-sense multiple access in heterogeneous wireless networks,” *IEEE Transactions on Mobile Com-*

puting, vol. 20, no. 9, pp. 2848–2861, 2021.

[164] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[165] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[166] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[167] X. Wang, J. Li, X. Kuang, Y.-a. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *Journal of Parallel and Distributed Computing*, vol. 130, pp. 12–23, 2019.

[168] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Computer Science Review*, vol. 34, p. 100199, 2019.

[169] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[170] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[171] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018.

[172] A. Kurakin, I. J. Goodfellow, and S. Bengio, "adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[173] A. Haydari, M. Zhang, and C.-N. Chuah, “Adversarial attacks and defense in deep reinforcement learning (drl)-based traffic signal controllers,” *IEEE Open Journal of Intelligent Transportation Systems*, 2021.

[174] L. Hussenot, M. Geist, and O. Pietquin, “Manipulating neural policies with adversarial observations,” in *Real-world Sequential Decision Making Workshop, ICML*, 2019.

[175] Z. Yuan, J. Zhang, Y. Jia, C. Tan, T. Xue, and S. Shan, “Meta gradient adversarial attack,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7748–7757, 2021.

[176] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, “On detecting adversarial perturbations,” *arXiv preprint arXiv:1702.04267*, 2017.

[177] Y. Dong, H. Su, J. Zhu, and F. Bao, “Towards interpretable deep neural networks by leveraging adversarial examples,” *arXiv preprint arXiv:1708.05493*, 2017.

[178] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” *arXiv preprint arXiv:1705.07204*, 2017.

[179] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, IEEE, 2016.

[180] R. L. Whited, Q. T. Swanquist, J. E. Shipman, and J. R. Moon Jr, “Out of control: The (over) use of controls in accounting research,” *The Accounting Review*, vol. 97, no. 3, pp. 395–413, 2022.

[181] Y. Zhang, X. Shi, H. Zhang, Y. Cao, and V. Terzija, “Review on deep learning applications in frequency analysis and control of modern power system,” *International Journal of Electrical Power & Energy Systems*, vol. 136, p. 107744, 2022.

[182] A. Visioli, *Practical PID control.* Springer Science & Business Media, 2006.

[183] P. Shah and S. Agashe, "Review of fractional pid controller," *Mechatronics*, vol. 38, pp. 29–41, 2016.

[184] W. J. Shipman and L. C. Coetzee, "Reinforcement learning and deep neural networks for pi controller tuning," *IFAC-PapersOnLine*, vol. 52, no. 14, pp. 111–116, 2019.

[185] P. C. Sahu, R. Baliarsingh, R. C. Prusty, and S. Panda, "Automatic generation control of diverse energy source-based multiarea power system under deep q-network-based fuzzy-t2 controller," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, pp. 1–22, 2020.

[186] P. C. Sahu, R. Baliarsingh, R. C. Prusty, and S. Panda, "Novel dqn optimised tilt fuzzy cascade controller for frequency stability of a tidal energy-based ac microgrid," *International Journal of Ambient Energy*, pp. 1–13, 2020.

[187] Y. Zheng, Q. Sun, Z. Chen, M. Sun, J. Tao, and H. Sun, "Deep q-network based real-time active disturbance rejection controller parameter tuning for multi-area interconnected power systems," *Neurocomputing*, vol. 460, pp. 360–373, 2021.

[188] M. Yiming, P. Boyu, L. Gongqing, L. Yongwen, and Z. Deliang, "Feedforward feedback control based on dqn," in *2020 Chinese Control and Decision Conference (CCDC)*, pp. 550–554, IEEE, 2020.

[189] T. Zhou, D. Tang, H. Zhu, and Z. Zhang, "Multi-agent reinforcement learning for online scheduling in smart factories," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102202, 2021.

[190] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[191] O. Katsuhiko, *Modern Control Engineering*. Pearson, 2009.

[192] R. C. Dorf and R. H. Bishop, *Modern control Systems, 13th edition*. Pearson, Hoboken, New Jersey, 2017.

[193] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[194] J. Brownlee, *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[195] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, p. 22, 2019.

[196] N. A. F. Sheet, "Intelligent flight control systems," *NASA Dryden Flight Research Center*, 2014.

[197] H. Bou-Ammar, H. Voos, and W. Ertel, "Controller design for quadrotor uavs using reinforcement learning," in *2010 IEEE International Conference on Control Applications*, pp. 2130–2135, IEEE, 2010.

[198] A. Plaat, "Deep reinforcement learning," *arXiv preprint arXiv:2201.02135*, 2022.

[199] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi, "Autonomous quadrotor landing using deep reinforcement learning," *arXiv preprint arXiv:1709.03339*, 2017.

[200] M. A. Wiering, M. Withagen, and M. M. Drugan, "Model-based multi-objective reinforcement learning," in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–6, IEEE, 2014.

[201] H. Kamalzadeh and M. Hahsler, "Pomdp: Introduction to partially observable markov decision processes," *Tekn. rapport*, 2019.

[202] F. Alkhateeb, E. Al Maghayreh, and S. Aljawarneh, "A multi agent-based system for securing university campus: Design and architecture," in *2010 International Conference on Intelligent Systems, Modelling and Simulation*, pp. 75–79, IEEE, 2010.

[203] Z. Huang, "Consensus control of multiple-quadcopter systems under communication delays," Master's thesis, Dalhousie University, 2017.

[204] S. H. Chan, *Adversarial Attack*. Purdue University, 2018.

[205] S. Zhou, Z. Hu, W. Gu, M. Jiang, M. Chen, Q. Hong, and C. Booth, "Combined heat and power system intelligent economic dispatch: A deep reinforcement learning approach," *International Journal of Electrical Power & Energy Systems*, vol. 120, p. 106016, 2020.

[206] K. C. Kosaraju, M. Figura, and V. Gupta, "Adversarial multi-agent reinforcement learning (adv-marl)." `https://github.com/asokraju/adv-marl`, 2020.

[207] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*, vol. 33. Princeton University Press, 2010.

[208] S. Wang, R. Diao, T. Lan, Z. Wang, D. Shi, H. Li, and X. Lu, "A drl-aided multi-layer stability model calibration platform considering multiple events," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2020.

[209] Y. Liu, B. Cao, and H. Li, "Improving ant colony optimization algorithm with epsilon greedy and levy flight," *Complex & Intelligent Systems*, vol. 7, no. 4, pp. 1711–1722, 2021.

[210] S. Ohnishi, E. Uchibe, Y. Yamaguchi, K. Nakanishi, Y. Yasui, and S. Ishii, "Constrained deep q-learning gradually approaching ordinary q-learning," *Frontiers in Neurorobotics*, vol. 13, p. 103, 2019.

[211] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[212] K. Ammouri, "Deep reinforcement learning for temperature control in buildings and adversarial attacks," Master's thesis, School of Electrical Engineering and Computer Science (EECS), 2021.

[213] Y. Yu, "Cs-dlma." `https://github.com/YidingYu/CS-DLMA`, 2019.

[214] N. Abramson, "The aloha system: Another alternative for computer communications," in *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, pp. 281–285, 1970.

[215] F. F. Kuo, "The aloha system," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 41–44, 1995.

[216] F. F. Kuo, "Computer networks–the aloha system.," tech. rep., Hawaii Univ at Manoa Honolulu Dept of Electrical Engineering, 1981.

[217] P. Jung, "Time division multiple access (tdma)," *Wiley Encyclopedia of Telecommunications*, 2003.

[218] Y. Luo and J. Pang, "Observer-based event-triggered finite-time consensus for general linear leader-follower multi-agent systems," *Advances in Continuous and Discrete Models*, vol. 2022, no. 1, pp. 1–17, 2022.

Appendices

Appendix A

Data Transmission in 2-D and 3-D Spaces: Various Activation Functions

Average reward and loss convergence using various activation functions in the structure of DQN agent NN for time-delayed data transmission related to Chapters 6 and 7 in 2-D and 3-D spaces are demonstrated in this Appendix. The time-delayed data transmission occurs between agents of a leader-follower MADRL system containing $N = 5$ agents and $P = 3$ various clusters. Various activation functions, including rectified linear unit (ReLU), rectified linear unit 6 (ReLU6), exponential linear unit (ELU), scaled exponential linear units (SELU), and Swish activation functions, are integrated with the MSE loss function. The following results are obtained in 150000 time steps. The five agents' two-dimensional (2-D space) positions are

$$(x^i, y^i) = \{(0.1, 0.22), (0.15, 0.27), (0.30, 0.9), (0.35, 0.14), (0.41, 0.40)\}.$$

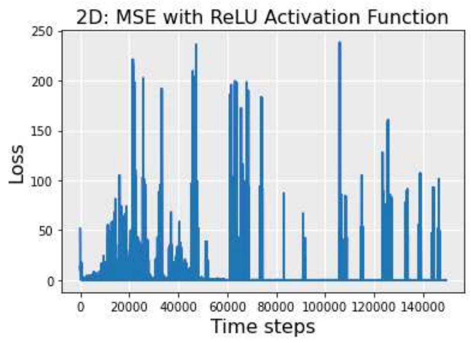
Furthermore, the five agents' three-dimensional (3-D space) positions are

$$(x^i, y^i, z^i) = \{(0.1, 0.22, 0.40), (0.15, 0.27, 0.14), (0.30, 0.9, 0.20), (0.35, 0.14, 0.50), (0.41, 0.40, 0.28)\}.$$

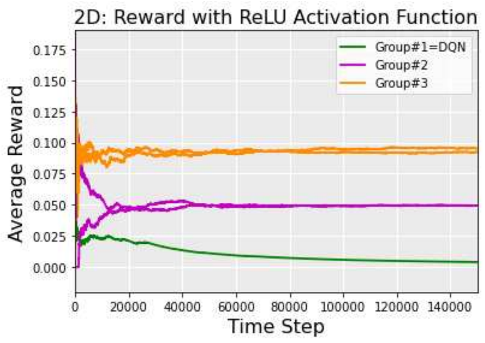
A.1 Rectified Linear Unit (ReLU) Activation Function

$$ReLU(z) = \max(0, z) \tag{A.1}$$

A.1.1 2-D Space



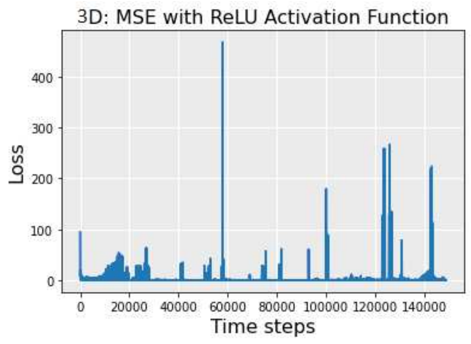
(a) Average loss convergence, ReLU.



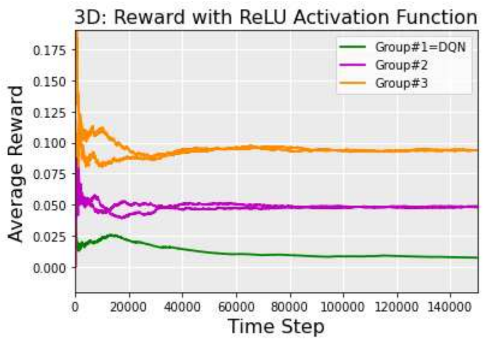
(b) Average reward convergence, ReLU.

Figure A.1: Average reward and loss convergence of data transmission in MADRL system using ReLU activation function in 2-D space.

A.1.2 3-D Space



(a) Average loss convergence, ReLU.



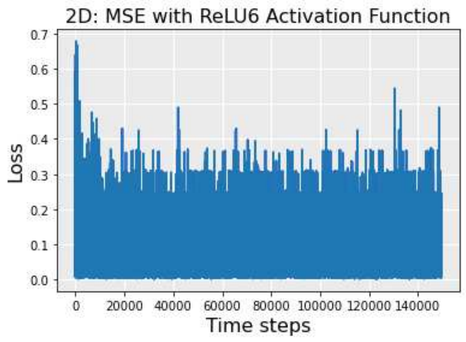
(b) Average reward convergence, ReLU.

Figure A.2: Average reward and loss convergence of data transmission in MADRL system using ReLU activation function in 3-D space.

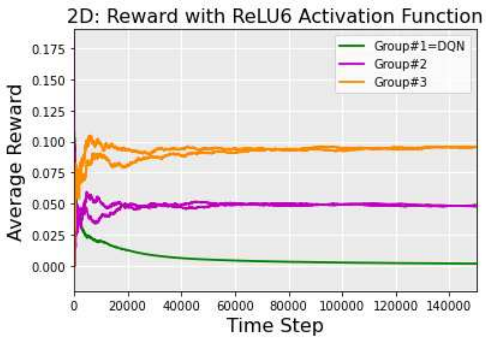
A.2 Rectified Linear Unit 6 (ReLU6) Activation Function

$$\text{ReLU6}(z) = \min(\max(0, z), 6) \tag{A.2}$$

A.2.1 2-D Space



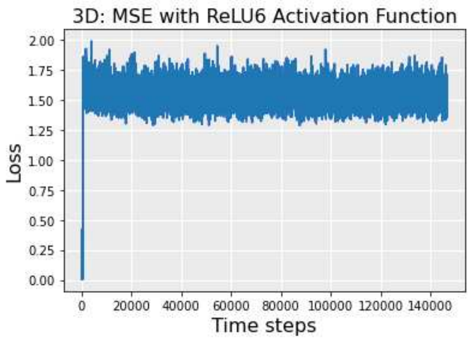
(a) Average loss convergence, ReLU6.



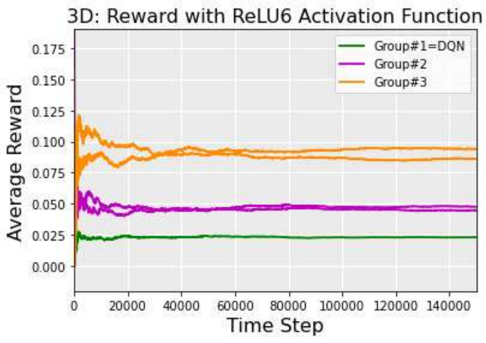
(b) Average reward convergence, ReLU6.

Figure A.3: Average reward and loss convergence of data transmission in MADRL system using ReLU6 activation function in 2-D space.

A.2.2 3-D Space



(a) Average loss convergence, ReLU6.



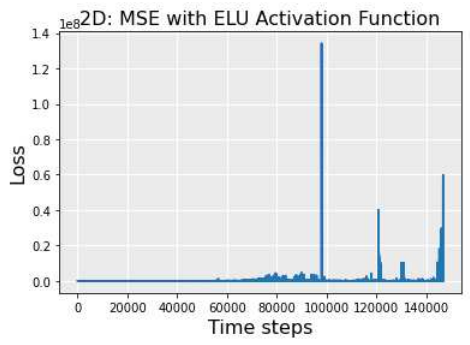
(b) Average reward convergence, ReLU6.

Figure A.4: Average reward and loss convergence of data transmission in MADRL system using ReLU6 activation function in 3-D space.

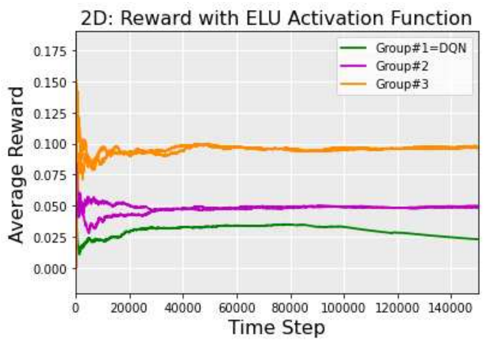
A.3 Exponential Linear Unit (ELU) Activation Function

$$ELU(z) = z \text{ if } z \geq 0$$
$$ELU(z) = \alpha(\exp(z) - 1) \text{ if } z < 0 \tag{A.3}$$

A.3.1 2-D Space



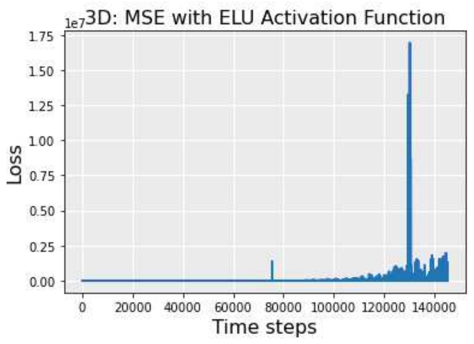
(a) Average loss convergence, ELU.



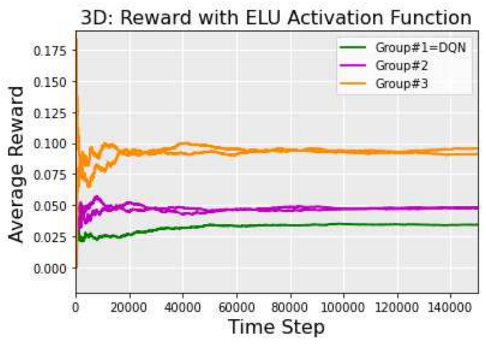
(b) Average reward convergence, ELU.

Figure A.5: Average reward and loss convergence of data transmission in MADRL system using ELU activation function in 2-D space.

A.3.2 3-D Space



(a) Average loss convergence, ELU.



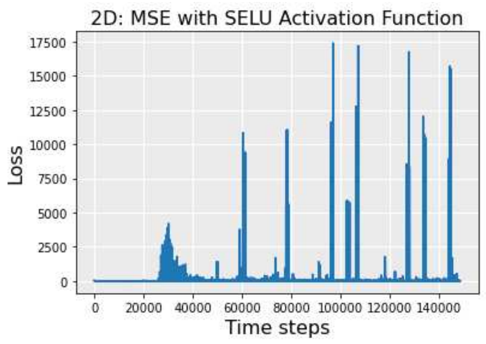
(b) Average reward convergence, ELU.

Figure A.6: Average reward and loss convergence of data transmission in MADRL system using ELU activation function in 3-D space.

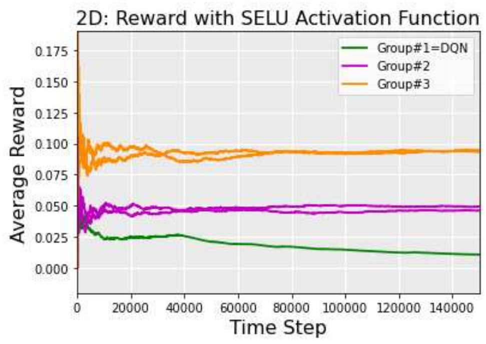
A.4 Scaled Exponential Linear Units (SELU) Activation Function

$$SELU(z) = \lambda z \text{ if } z \geq 0$$
$$SELU(z) = \lambda \alpha(\exp(z) - 1) \text{ if } z < 0 \tag{A.4}$$

A.4.1 2-D Space



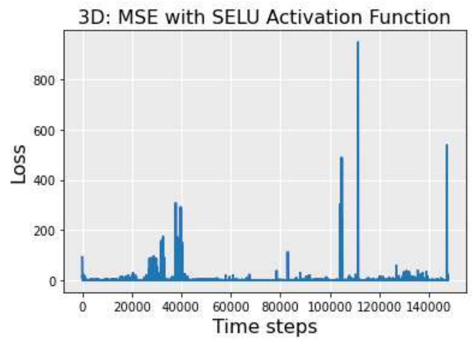
(a) Average loss convergence, SELU.



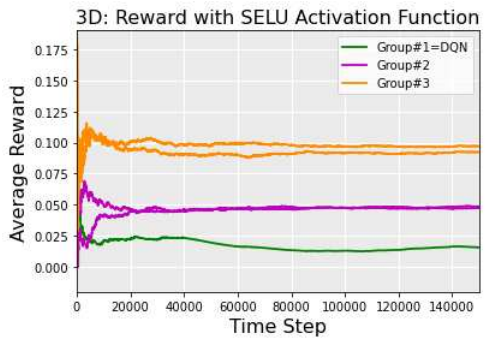
(b) Average reward convergence, SELU.

Figure A.7: Average reward and loss convergence of data transmission in MADRL system using SELU activation function in 2-D space.

A.4.2 3-D Space



(a) Average loss convergence, SELU.



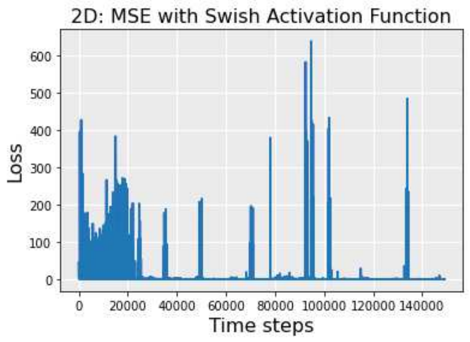
(b) Average reward convergence, SELU.

Figure A.8: Average reward and loss convergence of data transmission in MADRL system using SELU activation function in 3-D space.

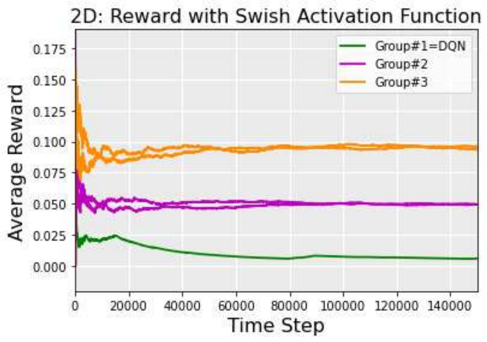
A.5 Swish Activation Function

$$\begin{aligned} \text{Swish}(z) &= z * \text{Sigmoid}(z) \\ &= \frac{z}{1+e^{-z}} \end{aligned} \tag{A.5}$$

A.5.1 2-D space



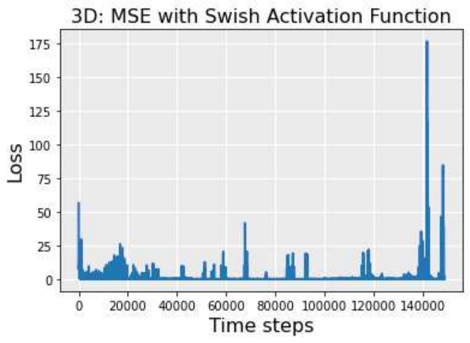
(a) Average loss convergence, Swish.



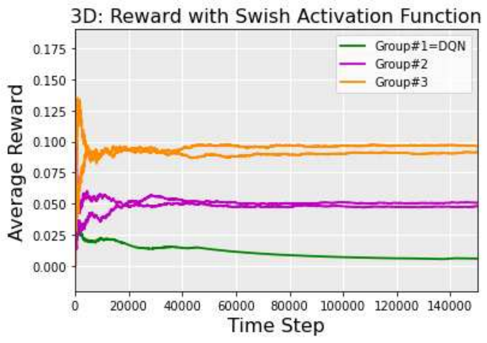
(b) Average reward convergence, Swish.

Figure A.9: Average reward and loss convergence of data transmission in MADRL system using Swish activation function in 2-D space.

A.5.2 3-D space



(a) Average loss convergence, Swish.



(b) Average reward convergence, Swish.

Figure A.10: Average reward and loss convergence of data transmission in MADRL system using Swish activation function in 3-D space.

Table A.1: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 150000 time steps by considering time-delay and distance-based reward using ReLU activation function in 2-D and 3-D spaces.

Rewards and Loss / Activation Function	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
ReLU (2-D Space)	0.0	0.0737	0.0390	0.1302	0.0812	0.3243	1.4514
ReLU (3-D Space)	0.0	0.6680	0.0287	0.0952	0.1003	0.2912	0.6799

Table A.2: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 150000 time steps by considering time-delay and distance-based reward using ReLU6 activation function in 2-D and 3-D spaces.

Rewards and Loss / Activation Function	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
ReLU6 (2-D Space)	0.0	0.0245	0.0879	0.1013	0.1195	0.3333	0.0713
ReLU6 (3-D Space)	0.0298	0.0382	0.0431	0.0619	0.0764	0.2496	1.5150

Table A.3: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 150000 time steps by considering time-delay and distance-based reward using ELU activation function in 2-D and 3-D spaces.

Rewards and Loss / Activation Function	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
ELU (2-D Space)	0.0008	0.0638	0.0390	0.0965	0.0764	0.2767	55587.9454
ELU (3-D Space)	0.0317	0.0286	0.0623	0.1095	0.1147	0.3470	11100.7798

Table A.4: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 150000 time steps by considering time-delay and distance-based reward using SELU activation function in 2-D and 3-D spaces.

Rewards and Loss / Activation Function	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
SELU (2-D Space)	0.0072	0.0688	0.0537	0.0868	0.0908	0.3074	39.4038
SELU (3-D Space)	0.0073	0.0525	0.0479	0.0810	0.0812	0.2701	0.7636

Table A.5: Comparison of each agent's average reward and DQN loss of a heterogeneous MAS, including $N = 5$ agents in $P = 3$ different clusters, during 150000 time steps by considering time-delay and distance-based reward using Swish activation function in 2-D and 3-D spaces.

Rewards and Loss / Activation Function	Agent 1 Reward	Agent 2 Reward	Agent 3 Reward	Agent 4 Reward	Agent 5 Reward	Team Reward	DQN Loss
Swish (2-D Space)	0.0260	0.0540	0.0341	0.0579	0.1051	0.2774	2.6988
Swish (3-D Space)	0.0002	0.0382	0.0719	0.1048	0.0812	0.2964	0.3429