COOPERATIVE MULTI AGENTS FOR INTELLIGENT INTRUSION DETECTION AND PREVENTION SYSTEMS

SHAHABODDIN SHAMSHIRBAND

THESIS SUBMITTED IN FULFILMENT

OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY UNIVERSITY OF MALAYA

KUALA LUMPUR

2014

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Shahaboddin Shamshirband Passport No: H95621769

Registration/Matric No: WHA110016

Name of Degree: Doctor of Philosophy

Title of Thesis: COOPERATIVE MULTI AGENTS FOR INTELLIGENT INTRUSION DETECTION AND PREVENTION SYSTEMS

Field of Study: Intrusion Detection System

I do solemnly and sincerely declare that:

I am the sole author/writer of this Work;
This Work is original;
Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name: Miss Laiha Binti Mat Kiah
Designation: Associate Professor
Witness's Signature Date

Name: Nor Badrul Anuar Bin Jumaat
Designation: Senior Lecturer

Abstract

Owing to the distributed nature of modern attacks (e.g. denial-of-service), it is extremely challenging to detect such malicious behaviour using traditional intrusion detection systems. In this thesis, we investigate the possibility of adapting an intelligent system to an Intrusion Detection System (IDS) by proposing a cooperative and intelligent detection and prevention system using machine learning approaches, and aim to facilitate the detection and prevention process in a distributed environment. Firstly, we review the state of the art of intelligent intrusion detection and prevention system (IIDPS), and highlight the security requirement of cooperative based-IIDPS. Adaptive optimization techniques such as fuzzy logic controller (FLC), reinforcement learning are discussed in this thesis in order to adopt Q-leaning algorithm to FLCs. We investigate the detection capability based on the fuzzy Q-learning (FQL) algorithm and evaluate it using distribute denial of service attacks (DDoS). Later, we investigate the game based-FQL algorithm by combining the game theoretic approach and the fuzzy Q-learning algorithm. This thesis evaluates the proposed solution using flooding attacks in wireless sensor networks (i.e. a type of DDoS attack). In order to measure the evaluation, several performance metrics, such as frequency of convergence of the detection scheme, accuracy of detection, false alarm rate, defence rate and energy consumption, are addressed as part of detection and prevention scheme. We perform the aforementioned investigations using several simulation experiments. The quantitative results acquired from the experiments are benchmarked with corresponding results acquired from the cooperative attack detection scheme. Through the result comparisons, we demonstrate the significance of cooperative detection mechanism, for detecting distributed denial of service attacks in a timely and energy-efficient manner, accuracy of detection and defence, as well as false alarm rate.

Abstrak

Disebabkan oleh serangan moden yang bersifat teragih (cth: nafi khidmat), ianya amat mencabar untuk mengesan tingkah laku hasad dengan menggunakan sistem pengesanan pencerobohan tradisional. Dalam tesis ini, kami menyiasat kebarangkalian untuk menyesuaikan satu sistem pintar pada satu Sistem Pengesanan Pencerobohan (IDS) dengan mencadangkan satu Sistem Pengesanan dan Pencengahan bersifat kerjasama serta pintar menggunakan kaedah pembelajaran mesin, dan bertujuan untuk memudahkan proses pengesanan dan pencegahan dalam persekitaran yang teragih. Pertama, kami mengkaji perkembangan terkini dalam sistem pengesan dan pencegahan pencerobohan pintar (IIDPS), dan menggariskan keperluan keselamatan bagi IIDPS bersifat kerjasasama. Teknik penyesuaian optimum seperti fuzzy logic controller (FLC) dan reinforcement learning dibincangkan dalam tesis ini bagi tujuan penyesuaian algoritma Q-Learning pada FLCs. Kami menyiasat keupayaan pengesanan berasaskan algoritma fuzzy Q-learning (FQL) dan menilainya dengan mengunakan serangan nafi khidmat teragih (DDoS). Seterusnya, kami menyiast algoritma game based-FQL dengan menggabungkan kaedah game theory dan algoritma fuzzy Q-learning. Tesis ini menilai cadangan penyelesaian dengan menggunakan serangan flooding (i.e. satu jenis serangan nafi khidmat) dalam rangkaian sensor tanpa wayar. Untuk tujuan mengukur penilaian, beberapa metrik prestasi, seperti kekerapan penumpuan skim pengesanan, ketepatan pengesanan, kadar penggera palsu, kadar pertahanan dan penggunaan tenaga, ditangani sebahagian daripada skim pengesanan dan pencegahan. Kami melaksanakan menilaian yang dinyatakan dengan mengunakan beberapa ujikaji simulasi. Segala keputusan kuantitatif hasil dari ujikaji-ujikasi ditanda-araskan dengan hasil keputusan yang diperolehi daripada skim kerjasama pengesanan serangan yang lain. Melalui perbandingan hasil keputusan, kami menunjukkan kepentingan mekanisme pengesanan kerjasama, dalam mengesan serangan nafi khidmat teragih di dalam waktu yang tepat

dan penggunaan tenaga yang berkesan, ketepatan dalam mengesan dan pertahanan, dan juga kadar nilai penggera palsu.

Acknowledgement

Allah is very kind, merciful and sympathetic. His benevolence and blessings enable me to accomplish this thesis.

Firstly, I would like to express my profound gratitude to my supervisor, Prof. Dr. Miss Laiha Binti Mat Kiah, Dr. Nor Badrul Anuar Bin Jumaat, for giving me the opportunity to work with them, for encouraging me to be self-motivated and ambitious about my career, for their support in crucial moments and for assisting me in writing papers and reviews. I especially appreciate their hard-working attitude.

I must also acknowledge the financial support given by the financial support of the Bright Sparks Program at University of Malaya, and Islamic Azad University, Chalous Branch, Mazandaran, Iran, which have been equally important to make possible this thesis and allowed me to present results and exchange knowledge and skills in conferences, journals, workshops and research stays abroad.

I am grateful to the members of the Faculty of Computer Science and Information Technology in University of Malaya for their tremendous guidance and continuous motivational and technical support. Finally, I would like to strongly thank my wife and parents for the support they have provided me through my entire life.

Contents

ABSTRACT III
ABSTRAK IV
ACKNOWLEDGEMENT VI
CONTENTS VII
LIST OF FIGURES IX
LIST OF TABLES X
ACRONYMS AND ABBREVIATIONS XI

CHAPTER 1 : INTRODUCTION 1

1.1 OVERVIEW 1

1.2 INTRUSION DETECTION SYSTEMS 1

1.3 RESEARCH MOTIVATION 3

1.4 RESEARCH METHODOLOGY 4

1.5 THESIS OUTLINES 7

CHAPTER 2 : INTELLIGENT INTRUSION DETECTION AND PREVENTION SYSTEM 9

2.1 OVERVIEW 9

2.1.1 DATA SET 16

2.2 STATE-OF-THE-ART TAI, CI AND MCI IN IDPS 18

2.2.1 TRADITIONAL ARTIFICIAL INTELLIGENCE (TAI) 18
2.2.2 COMPUTATIONAL INTELLIGENCE (CI) 21
2.2.3 MULTI AGENT-BASED COMPUTATIONAL INTELLIGENCE (MCI) 24

2.3 DETECTION RATES PERFORMANCE METHOD AND CRITERIA FOR EVALUATION 28

2.3.1 TRADITIONAL ARTIFICIAL INTELLIGENCE 35
2.3.2 COMPUTATIONAL INTELLIGENCE 38
2.3.3 MULTI AGENT SYSTEM-BASED COMPUTATIONAL INTELLIGENCE 41
2.3.4 COMPARATIVE DISCUSSION OF DETECTION RATE EVALUATION 43

2.4 INTELLIGENT INTRUSION DETECTION AND PREVENTION SYSTEM (IIDPS) 47

2.5 DISCUSSION 51

2.6 CHAPTER SUMMARY 54

CHAPTER 3 : ADAPTIVE OPTIMIZATION TECHNIQUES 56

3.1 OVERVIEW 56

3.2 FUZZY LOGIC 57

3.2.1 FUZZY LOGIC CONTROLLER DESIGN 60

3.3 REINFORCEMENT LEARNING 73

3.4 Q-LEARNING ADAPTATION TO FLCS 82

3.5 ADAPTATION OF MULTI-AGENT BASED FUZZY REINFORCEMENT LEARNING 86

3.6 GAME THEORY 88

3.7 COLLABORATIVE-IIDPS ARCHITECTURE 97

3.8 DISCUSSION 102

3.9 CHAPTER SUMMARY 103

CHAPTER 4 : GAME THEORETIC APPROACH USING FUZZY Q-LEARNING 105

4.1 PROPOSED MODEL 106

4.1.1 WSN MODEL 106
4.1.2 METHODOLOGIES AND TECHNIQUES USED 108
4.1.3 POSSIBLE ATTACK CATEGORIES 110

4.2 THE ARCHITECTURE OF COOPERATIVE GAME-BASED FQL IDPS 111

4.2.1 GAME DESIGN 112
4.2.2 FUZZY Q-LEARNING ALGORITHM 121

4.3 CHAPTER SUMMARY 124

CHAPTER 5 : FRAMEWORK EVALUATION 125

5.1 SIMULATION AND ANALYSIS 126

5.1.1 GENERAL TOOLS 126
5.1.2 DESIGN ASSUMPTIONS 126
5.1.3 SIMULATION SETUP 127
5.1.4 GENERATING AND ANALYSING THE FLOOD ATTACK STRATEGY 129
5.1.5 ANALYSIS OF THE GAME-BASED FQL IDPS IN TERMS OF DETECTION ACCURACY 130
5.1.6 ANALYSIS OF GAME-BASED FQL IDPS IN TERMS OF DEFENSE RATE 132
5.1.7 ANALYSIS OF GAME-BASED FQL IDPS IN TERMS OF NUMBER OF LIVE NODES 133
5.1.8 ANALYSIS OF GAME-BASED FQL IDPS IN TERMS OF ENERGY CONSUMPTION OVER TIME 135
5.1.9 ANALYSIS OF THE ENERGY CONSUMED BY DIFFERENT DEPLOYED NODES IN THE GAME-FQL 136
5.1.10 ANALYSIS OF THE COMPUTATIONAL TIME IN THE GAME-FQL 137

5.2 CHAPTER SUMMARY 138

CHAPTER 6 : CONCLUSION 139

6.1 ACHIEVEMENTS OF THE STUDY 139

6.2 LIMITATIONS OF THE STUDY 144

6.3 FUTURE WORK 145

List of Figures

Figure 1.1: Research Methodologies 5
Figure 2.1: Chronological order of [TAI], [CI] and [MCI] based IIDPS 27
Figure 2.2: Year wise distribution of articles for the various types of classifier layouts 35
Figure 2.3: Comparison of TAI methods in terms of detection rate and false alarm rate 38
Figure 2.4: Comparison of CI methods in terms of detection rate and false alarm rate 41
Figure 2.5: Comparison of Multi agent based CI methods in terms of detection rate and false alarm rate 43
Figure 2.6: General comparison of detection rate 46
Figure 2.7: General comparison of false alarm rate 47
Figure 2.8: Intelligent Intrusion Detection and Prevention architecture for networks 48
Figure 2.9: Tree plan classification of the anomaly-based IIDPS detection techniques 50
Figure 3.1: Block diagram of an FLC 61
Figure 3.2: The membership functions of linguistic variables for attack data source Tr 62
Figure 3.3: Example of fuzzification process 63
Figure 3.4: Membership functions of the input fuzzy sets as example 70
Figure 3.5: Basic FLC operation example 71
Figure 3.6: The basic elements of an RL problem 75
Figure 3.7: Basic scheme of generalized policy iteration 81
Figure 3.8: Multiple-agents acting in the same environment 87
Figure 3.9: Combination of NIDPS and HIDPS in a distributed Smart Grid Network (CIDPS) 97
Figure 3.10: Enhanced Collaborative-IIDPS functionality architecture within a network 99
Figure 4.1:A distributed hierarchical system perspective of a WSN 107
Figure 4.2: Model of a Cooperative Game-based IDPS and an attacker 109
Figure 4.3: Game-based defense system architecture 112
Figure 4.4: Block diagram of the FQL optimization system 121
Figure 5.1: Simulated WSN environment 128
Figure 5.2: Effects of UDP attack intensity on packet size 130
Figure 5.3: Victim node's energy level over time 130
Figure 5.4: Comparison of detection accuracy values 132
Figure 5.5: Game-based FQL in terms of accuracy of defense rate under attack trends 133
Figure 5.6: Number of live sensor nodes during simulation runtime (ms) 134
Figure 5.7: Total energy consumption versus number of sensor nodes under malicious attack 135
Figure 5.8:Total energy consumption versus number of sensors deployed in a network 136

List of Tables

Table 2.1: Total number of anomaly-based classifiers in Network based- IDPS 13
Table 2.2: Classification of the Datasets for NIDS and WIDS 17
Table 2.3: Possible status for an IIDPS reaction 28
Table 2.4: Evaluation metrics proposed by authors 29
Table 2.5: Classifications and comparisons of various intrusion detection approaches 31
Table 2.6: Classification of Traditional Artificial Intelligence-based IDPS 35
Table 2.7: Classification of computational intelligence in IDPS 39
Table 2.8: Classification of multi agent computational intelligence-based IDPS 41
Table 2.9: Panoramic comparison of evaluations on popular detection techniques. 44
Table 2.10: Vertical evaluation of technique categories 44
Table 2.11: Fundamentals of the anomaly-based IIDPS techniques 50
Table 2.12: Proposed Co-IIDPSs in terms of MCI methods classified according to our taxonomy 52
Table 2.13: The developed Co-IIDPSs (MCI) which met our proposed performance requirements 54
Table 3.1: Fuzzy rating for the occurrence of attack traffic 62
Table 3.2: Examples of 2-player, 2-action games. 90
Table 3.3: Examples of 2-player, 3-action games. 92
Table 4.1: Classification of Denial-of-Service attacks and defence at each protocol layer 110
Table 4.2: Game play between a sink node (IDS1) and an attacker 113
Table 4.3: Game play between a base station (IDS2) and an attacker 114
Table 4.4: The payoff matrix and utility functions 114
Table 4.5: Notations associated with the reward functions of a sink node and base station 117
Table 4.6: Utility function parameters 120
Table 4.7: Linguistic variables for fuzzy set input and output 123
Table 5.1: Wireless sensor network parameters in NS-2 128
Table 5.2: Simulation results of the detection algorithm for DDoS attacks 131
Table 5.3 : Performance comparison of G-FQL in terms of consuming time 137

Acronyms and Abbreviations

AA	analyser agent
ACC	ant colony clustering
ACO	ant colony optimization
AI	artificial intelligence
AIS	artificial immune system
BDI	beliefs–desires–intentions
BNF	backus–Naurform
CBR	case base reasoning
CI	computational intelligence
CIDPS	cloud-based intrusion detection and preventions system
CIDPS	collaborative intrusion detection and prevention system
CIIDS	computational intelligence intrusion detection systems
Co-IIDPS	collaborative-based intelligent IDPS
DA	decision agent
DC	data collector
DR	detection rate
EA	executor agent
FAR	false alarm rate
FCM	fuzzy C-mean
FCS	fuzzy classifier system
FN	false negatives
FP	false positives
FRB	fuzzy rule base
FRL	fuzzy reinforcement learning
FRLM	fuzzy reinforcement learning management
GA	genetic algorithm
GF	genetic fuzzy
HIDPS	host-based IDPS
HMM	hidden Markov model

HSN	hybrid sensor network
IDE	intrusion detection engine
IDRS	intrusion detection and response system
IDS	intrusion detection systems
IE	inference engine
JCR	journal citation reports
KM	knowledge management
MAM	multi-agent management
MANET	mobile ad-hoc networks
MAS	multi-agent system
MCI	multi-agent-computational intelligence
MDP	Marko decision process
MFF	multi layered feed-forward
ML	machine learning
MLP	multi-layer perceptron
NF	neuro-fuzzy
NIDES	next-generation intrusion detection expert system
NIDPS	network intrusion detection and prevention system
NSL-KDD	network services location-KDD
PSO	particle swarm optimization
RL	reinforcement learning
SC	soft computing
SGN	smart grid networks
SI	swarm intelligence
SOM	self-organizing map
SVM	support vector machine
TAI	traditional artificial intelligence
TD	temporal difference
TN	true negatives
TP	true positives
WIDPS	wireless intrusion detection and prevention system

WSN	wireless sensor network
GT	game theory

Chapter 1 : INTRODUCTION

1.1 Overview

The level of asset in Internet is positively correlated with network security. The hardness of ever-changing threat environment seems far from surrounded (Anuar et al., 2012). The recent statistics for the last three years show the battle against attacks.

According to Kaspersky Security Network (KSN) reported, in 2013 Kaspersky Lab products neutralized 5188740554 cyber-attacks on user computers and mobile devices. In terms of the mobile operating systems that are being targeted by malware, Android is still the number one target, attracting a whopping 98.05% of known malware (Kaspersky, 2013). In order to conduct all attacks over the Internet in 2013, cybercriminals used 10,604,273 unique hosts, which is 60.5% more than in 2012.

In 2013, a report by Gartner (2013) reveals that a sophisticated class of distributed denial of service (DDoS) attack sent an attack command to hundreds or even thousands of mobile agents, which then launched flooding attacks to access multiple websites.

A new report from Arbor's World-Wide Infrastructure Report (2012) shows that the size of distributed denial-of-service attacks have started to plateau, while application-layer and multi-vector attacks continue to evolve.

1.2 Intrusion Detection Systems

The security analysts use different approaches to analyse the threats, such as antivirus software, firewalls and Intrusion Detection Systems (IDSs). The use of an IDS or related system, such as an Intrusion Prevention System, is one of the most popular options in commercial due to their operation, openness and wide-acceptance as security

devices (Anuar et al., 2013). An intrusion detection system (IDS) and intrusion response system (IPS) operate to detect suspicious activities and respond to them.

There are hundreds of published works related to intrusion detection (Patcha et al., 2007), which aim to improve the efficiency and reliability of detection, prevention and response systems. Prevailing studies have so far focused on reducing alerts (Maggi et al., 2009), detecting DDoS attacks (Mirkovic et al., 2004), prioritizing incidents, eliminating and reducing false alarms (Steinberg et al., 2005), and increasing the self-reliance level of incident responses (Anuar et al., 2013). Artificial intelligence (AI) techniques attend to automate the intrusion detection and reduce human intervention. Intrusion detection system in artificial intelligent is categorized into three type: traditional artificial intelligence (TAI) and computational intelligence (CI) and multi agent-based CI (MCI) techniques that operate as classifiers.

In TAI techniques, network traffic activity is captured by single classifiers (i.e. fuzzy set, neural network, genetic algorithm and artificial immune system), thereafter, a profile representing its desired behavior is coded and finally a behavior model is created. Network events take place, the current profile is assigned and an anomaly score is computed by comparing the two behaviors. The score normally indicates the degree of irregularity for a specific event, such that the IDS raises a flag in the event an anomaly occurs when the score surpasses a certain threshold. Computational intelligence classifiers are meant to create an iterative process of observing patterns, adjusting to the mathematical form, and making predictions (Alpadin, 2010). MCI techniques function by applying the multi agent system to computational intelligence in order to enhance the performance of detection and response. On the other hand, cooperative multi agent system uses CI methods such as self organizing map (SOM), support vector machine (SVM), genetic algorithm (GA), reinforcement learning (RL) and game theory (GT) to determine temporal behavior and respond to any deviation.

The main objective of MCI consists of distributing multi agent system to each cluster to provide a CI mechanism that makes individual and cooperative decisions associated, for example, the use of this approach in Intrusion Detection Systems (Wooldridge, 2009). MCI has been widely employed in the domain of network security (W. Li et al., 2012) and cloud computing-based IDS (Doelitzscher et al., 2012). The main issue is to improve the accuracy of attack detection, false alarm rates as well as energy efficiency in the Intelligent Intrusion Detection and Prevention System (IIDPS), upon multi agent based computational intelligent IDPS in terms of Co-IIDPS.

1.3 Research Motivation

There are several classical security methodologies which have focused on particular types of attacks to prevent the attacks. An intelligent intrusion detection and prevenstion can be a line of defense. It is impossible, or even infeasible, to guarantee perfect prevention. Not all types of attacks are known and new ones appear constantly. As a result, attackers can always find security holes to exploit. For confident environments, it makes sense to establish a line of shield: An Intelligent Intrusion Detection and Prevention System (IIDPS) able to detect attacks and warn the sensors and the operator about it.

Most IIDPSs have focused on local detection in network, i.e., allowing nodes to locally detect specific attacks which are performed in their neighborhood (Ponomarchuk et al., 2010). Da Silva et al. (2005) propose a similar IDS systems, where they are able to monitor nodes in a network and responsible to observe other neighbors. They listen to messages in their radio range and store in a buffer specific message fields that might be useful to an IDS system running within a sensor node. Wang et al. (2006) focus on the detection of selfish nodes to preserve their resources at the expense of others. Loo et al. (2006) applied the IDSs for ad hoc networks. In all the above work,

there is no collaboration among the sensor nodes. The only collaborative approaches we are aware of focus on the local detection of intrusion detection based on traditional artificial intelligence (Patel et al., 2013).

More prevalent work has been done in intrusion detection for ad-hoc networks (Huang et al., 2013). In such networks, distributed and cooperative IDS architectures are also preferable. Detailed distributed designs, actual detection techniques and their performance have been studied in more depth. We are unaware of any work that has investigated the issue of distributed denial of service (DDoS) attack detection and response in a general collaborative way for networks. Thus, the lack of cooperative and disributed mechansim which utilizes computational intelligence have been our motivation for creating a game based cooperative IDPS to overcome the problem of accuracy of detection, response and false alarm rate.

1.4 Research Methodology

This section outlines the research methodology adopted in this thesis. The phases of the research methods are presented. The details of the methodology are explained in this section as shown in Figure 1.1. The literature review and problem statement are discussed in Phase 1. In Phase 2, the research objective is argued. The system designs are proposed in Phase 3. The evaluation and analysis are discussed in Phase 4.

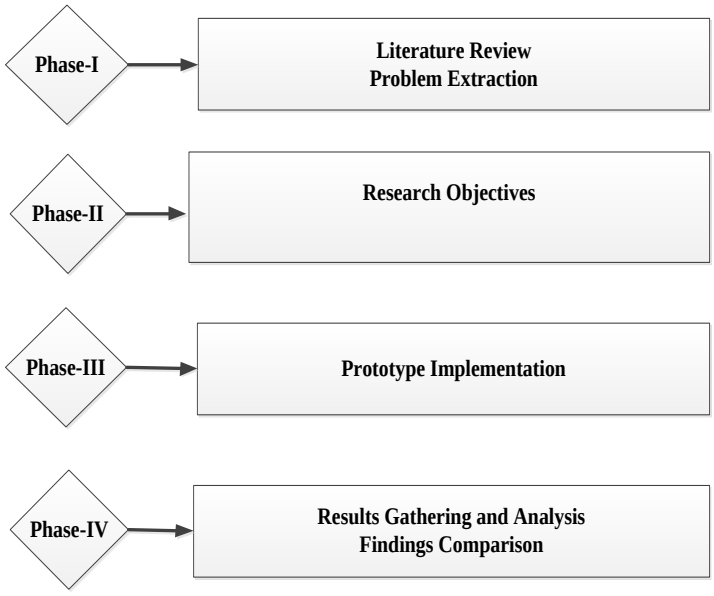

Figure 1.1: Research Methodologies

Phase I: Literature Review and Problem Extraction

The focus of this thesis is to design a cooperative intelligent intrusion detection and prevention security schemes. Therefore, the outline is based upon the following related works:

1) Initially, the existing intrusion detection schemes designed for network environment are categorized as **(a)** *data security schemes* or **(b)** *application security schemes.*

2) Afterwards, the existing data security schemes design for IDPSs are sub-categorized as **(a)** *traditional intelligence* **(b)** *computational intelligence and* **(c)** *multi agent based computational intelligence IDPS schemes.*

3) Thereafter, the selected real data schemes are critically analysed to identify the computation intensive operations and security issues that need to be addressed.

Phase II: Research Objectives

The aim of this thesis is to propose a novel cooperative multi agent intelligent intrusion detection and prevention scheme to address the intrusions. In order to achieve this aim, several issues need to be thoroughly understood, analysed and evaluated, as follows:

(a) To comprehensively investigate the domain of cooperative multi agent intelligent intrusion detection and response, and identify the key issues with respect to the effective defense against intrusions.

(b) To design and implement a novel cooperative IIDPS framework to facilitate a practical evaluation of intrusion detection.

(c) To evaluate the performance of a proposed framework in terms of accuracy of detection and false alarm rate by validating it using evaluation studies at different stages in order to demonstrate the progress of results.

Phase III: Prototype Implementation

1) The cooperative IDPS is developed in a network simulator.
2) To generate an attack with a random function, which selects subject nodes from each cluster to attack, the selected nodes adjust their functions to send flooding packets to the cluster head.
3) The game based IDS uses Low Energy Adaptive Clustering Hierarchy (LEACH) protocol in the simulation, as it closely reflects WSN in practice and is also capable of dealing with energy consumption concerns in WSNs. The simulations were run for 1000s with LEACH as the routing protocol, the initial access point energy was 100 joules, the effective transmission range of the wireless radio for the access point was 100m, the sink node transmission range was 100m, and the common node transmission range was 50m.

Phase IV: Results and Comparison

We perform simulations to analyse the proposed scheme, for performance analysis using algorithmic and network-level parameters. The impact of security schemes on distributed denial of service attack are evaluated on the basis of accuracy of detection, accuracy of response, false alarm rate, time complexity, number of node alive, and energy consumption on network device while performing intrusion detection, intrusion response, and anomaly clustering operations.

1.5 Thesis outlines

The objectives presented above relate to the general sequence of the material presented in this thesis, the structure of which is discussed in five chapters.

Chapter 2 introduces a comprehensive taxonomy along with state-of-the-art intelligent intrusion detection and prevention systems (IIDPS), and specifically reviews their response capabilities in networks. The IIDPS were assessed and categorized into three trends: traditional artificial intelligence, computational intelligence and multi-agent-based computational.

Chapter 3 provides the details of the theoretical basis and the mathematical techniques appropriate for adaptive optimization techniques. We introduce the basic concepts and design of fuzzy logic controller, reinforcement learning and game theory. The chapter proposes an effective cooperative multi agent architecture based on computational intelligence methods for detection and prevention of attacks. This chapter highlights the advantages of such studies and discusses how they can be combined to produce a more effective means of detecting of intrusions.

Chapter 4 presents the main contribution of this thesis: a novel collaborative game based IDPS. In presenting the framework, this chapter begins by presenting the core foundation behind the framework as well as its operational characteristics. This chapter comprises the study by conducting multiple experiments to validate and evaluate the proposed game based fuzzy Q-learning IDPS framework. In addition, example scenarios are provided to demonstrate how the proposed framework operates, and how the network simulator interfaces can be used to assist security analysts in making a decision.

Chapter 5 demonstrates the progress of the results and the evaluation study presents the experimental results in four stages. The first stage aims to validate the Game based fuzzy Q-learning IDPS in terms of accuracy of detection by comparing its results to the existing machine learning methods such as fuzzy logic controller, Q-learning, and fuzzy Q-learning approaches. Based upon the initial results of the first experiment, the second stage aims to enhance detection rate by using collaborative game theory- IDPS in terms of Game based fuzzy Q-learning IDPS. The third stage investigates the effectiveness of the proposed Game FQL-IDPS in achieving two different goals: first it investigates the influence of proposed method in terms of energy consumption over time; and secondly it investigates the effectiveness of proposed method in terms of energy consumption for different deployed nodes. The fourth stage investigates the performance of the proposed framework by measuring the time complexity during detection process. This chapter also gives an in-depth discussion of the implications of applying the proposed framework in practice, underlining the advantages as well as the limitations.

Chapter 6 presents the main conclusions drawn from this thesis, highlights the principle achievements and limitations of the work, and makes suggestions for potential of further enhancements.

Chapter 2 : INTELLIGENT INTRUSION DETECTION AND PREVENTION SYSTEM

This chapter studies the intelligent intrusion detection and prevention system (IIDPS) schemes in networks and cloud computing. More specially, the categorization of IDPS schemes in terms of traditional artificial intelligence (TAI), computational intelligence (CI) and multi-agent CI (single cloud, collaborative cloud) are discussed. We highlight the benefits of multi-agent system based CI in terms of collaborative based IIDPS (Co-IIDPS) to attain high accuracy of attacks. The research areas and directions in developing and deploying CI based Co-IIDPSs is mentioned.

2.1 Overview

Unlike prevalent intrusion detection and prevention system, intelligent intrusion detection and prevention system not only aim to foster the supportive effectiveness of detection, for instance, with increased accuracy of detection and decreased false alarm rate, but also have cooperative intelligent approaches. An intelligent actuator help enhance the cooperative effort of IDPS to communicate while detecting anomalies in areas including health, warfare and environment monitoring (Akyildiz et al., 2002). For example, health monitoring models adopt IDS as extraordinary parts to continuously capture quantitative data from an enormous number of wearable body sensor networks for longer periods (Hanson et al., 2009). Hybrid Sensor Network (HSN) architecture employs MicaZ sensors for the battlefield, which are skilled in tracking live vocal and magnetic weapon signals generated by enemy forces (Bokareva et al., 2006). The most recent publication by Kapitanova et al. (2012) demonstrates how robust fuzzy logic is in event detection by monitoring smoke via temperature sensors attached throughout the home environment; fire ignition may thus be detected, making the relevance of sensor applications apparent.

The existing application designs for wireless networks afford greater flexibility in establishing communications and increase system automation, though lack in security and privacy (N. Li et al., 2009). The core weaknesses with these sensor nodes lie in the limited-resource devices, i.e. power and processing units. For this reason, vulnerability to various security threats is notably high. Meanwhile, an adversary possesses passive and active abilities. It may thus implicate sensor nodes through access to secret information such as keys stored in the compromised node in addition to the potential to eavesdrop and alter (e.g. replay, forge, modify and delete) exposed nodes behaviour (Schaffer et al., 2012).

In mitigating security complications, traditional security tactics like firewall and cryptography are alternative options to prevent external intruders. Nevertheless, they are impractical in completely averting network resources from increasingly sophisticated internal attacks (Chen et al., 2002). A different security approach incorporates Intrusion Detection and Prevention Systems (IDPSs) to detect and impede intrusion by impostors. An Intrusion Detection System or other similar ones (e.g. Intrusion Prevention System, Intrusion Response System) monitor network traffic to analyse and detect attacks (Anuar et al., 2012). Three detection methods employed are: misuse, anomaly, and the hybrid model--a blend of the first two Fuchsberger et al. (2005). A misuse-based system identifies known patterns by matching observed data using simple rules. For instance, Snort-Wireless runs its default rule settings to process all malicious events observed by the sensor and adopt intrusion detection techniques Lockhart et al. (2005).

Anomaly-based detection refers to the discovery of anomalous patterns in measurement data that do not conform to the expected behaviour (Curiac et al., 2012). According to Dutkevych et al. (2007), an anomaly-based solution averts intrusion in real time systems by analysing protocol-based attacks and multidimensional traffic. The hybrid detection approach boosts the capabilities of a current Intrusion Detection and

Prevention System (IDPS) by joining the two intelligent methods of misuse and anomaly Wang et al. (2011). Aydin et al. (2009) designed a hybrid IDS by incorporating the Packet Header Anomaly Detection (PHAD) and Network Traffic Anomaly Detection (NETAD) systems, which are anomaly-based IDSs with misuse-based IDS Snort. The key concept behind hybrid detection is that misuse identifies known attacks while anomaly discovers unknown attacks.

The traditional misuse detection approaches display high performance regarding correct detection of known attacks and false alarm rates but fail to detect unknown attacks (D. Anderson et al., 1995). Therefore, a traditional anomaly detection approach is considered an alternative to detect constantly changing unknown attack behaviour, but it may also exhibit high false positive results.

Artificial Intelligence (AI) techniques play a role in automating the intrusion detection process to diminish human intermediation. The intrusion detection process based upon traditional artificial intelligence (TAI) entails methods such as fuzzy set, neural networks, and evolutionary computing, which operate as classifiers for anomaly detection (Idris et al., 2005). Denning (1987), with a rule-based expert system for Intrusion Detection Systems (IDSs), aimed to improve detection performance. Although these rules apply cover-known patterns, they are unable to adapt to the attacks' pattern changes (e.g. attack polymorphs). To alleviate the problem of attack modifications, computational intelligence (CI) is considered a high-accuracy detection method to be used in constructing an intelligent detection model and to automatically identify inconsistent activities (Kulkarni et al., 2011). Agah et al. (2004) detected attacks with the game theory-based reinforcement learning algorithm. The result was greater safety, but the energy efficiency issues remain to be addressed.

Despite the limited agreement on the exact procedure of constructing anomaly classifiers based on TAI and CI to address safety, there is a broadly accepted view that the sections of CI are, neuro fuzzy, genetic fuzzy and machine learning. By joining the autonomous multi agent with the CI or TAI methods, a number of the previously identified weaknesses such as accuracy of detection, false alarm rate as well as energy efficiency may be confronted. Toosi et al. (2007) combined the following three soft computing algorithms: neural network, fuzzy rules and genetic algorithm to improve the decision result optimization.

According to the existing reviews for anomaly resource-based monitoring, IDPS systems are divided into two categories: Host-based IDPS (HIDPS) and Network Based IDPS (NIDPS) systems (J. A. Anderson et al., 1995; Sherif et al., 2002). NIDPS monitors network traffic, in particular network segments or devices, after which it analyses network and protocol conduct to identify suspicious activities. HIDPS observes all, or portions of, the dynamic behaviour and state of a computer system. Unlike NIDPS which dynamically inspects network packets, HIDPS detects programs' access and resources. HIDPS offers the advantage of being easy to deploy without affecting existing infrastructures as opposed to NIDS which detects attacks at the transport protocol layer by quick responses.

Through this chapter, we investigate the application of TAI, CI and MCI for identifying present research challenges in preparing an intelligent intrusion detection and prevention system (IIDPS). A survey presents the state-of-the-art in the field of IIDPS and highlights the central issues to be addressed.

Table 2.1 provides the number of literature works dealing with TAI, CI and MCI approaches. The list of articles is provided as a general overview of TAIs, CIs and MCIs in terms of their characteristics and current challenges encumbering intelligent IDPS

development in sensor networks. The table comprises 4 horizontal sections (TAI, CI, and MCI) and 3 vertical divisions defining detection classifier types, the authors' work titles and the works' objectives. Embedding security mechanisms such as identifying possible known/unknown vulnerabilities, predicting user behaviour, analysing and deterring individuals from violating security policies, are adopted into the network protocols to facilitate the development of efficient intrusion recognition and reaction systems.

Table 2.1: Total number of anomaly-based classifiers in Network based- IDPS

Type of classifier		**Authors**	**Title of paper**	**Objectives**
Traditional Artificial Intelligence(TAI)	Neural networks (NN)	Debar et al. (1992)	A neural network component for an intrusion detection system	Prediction known user behavior – Design an IDS structure
		Cannady et al. (1998)	Artificial neural networks for misuse detection	Identifying possible known vulnerabilities-IDS
		Zhang et al. (2000)	Intrusion detection in ad-hoc networks	Intrusion detection and response mechanisms in ad-hoc networks- (Developing of WIDS)
		Bivens et al. (2002)	Network-based intrusion detection using neural networks	Analyzing and deterring individuals from violating security policies- (Explore network based intrusion detection)
		Bankovic et al. (2011)	Improving security in WMNs with reputation systems and self-organizing maps	Detect and confine unknown attacks- (Design framework for intrusion detection in Mobile Networks)
		Li et al. (2012)	The method of network intrusion detection based on the neural network GCBP algorithm	Analyzing and detection of unknown data packets- NIDS
	Fuzzy sets (FS)	Dickerson et al. (2001)	Fuzzy intrusion detection	To assess malicious activity- (Developing of WIDS)
		Bridges et al. (2000)	Fuzzy data mining and genetic algorithms applied to intrusion detection	Developed an architecture for intrusion detection
		Liang et al. (2005)	Event detection in wireless sensor networks using fuzzy logic system	Identifying possible known vulnerabilities-WIDS
		Abraham et al. (2007)	D-SCIDS: Distributed soft computing intrusion detection system	Evaluates and models NIDS
		Jianhui et al. (2008)	A Fast Fuzzy Set Intrusion Detection Model	Deterring Intrusion- Design IDS
		Wang et al. (2009)	A Detection Method for Routing Attacks Based on Fuzzy C-means	Analyzing and detection anomaly- WIDS

			Clustering	
	Artificial Immune system (AIS)	Jungwon et al. (2001)	Towards an artificial immune system for network intrusion detection: An investigation of clona l selection with a negative selection operator	Investigation of IDS
		Ma et al. (2007)	SAID: A Self-Adaptive Intrusion Detection System	Designing an architecture of Intrusion Detection System (WIDS)
	Genetic algorithm (GA)	Khanna et al. (2006)	Self-Organization of Sensor Networks Using Genetic Algorithms	Development of IDPS
		Sevil Sen et al. (2011)	Evolutionary computation techniques for intrusion detection in mobile ad hoc networks	Explore the use of evolutionary computation techniques in WIDS
Computational Intelligence(CI)	Soft computing (SC)	Mohajerani et al. (2003)	NFIDS: a neuro-fuzzy intrusion detection system	Developed anomaly Intrusion Detection system
		Gomez et al. (2002)	Evolving fuzzy classifiers for intrusion detection	Proposes a technique of anomaly detection
		Chavan et al. (2004)	Adaptive neuro-fuzzy intrusion detection systems	Design an intrusion detection system
		Toosi et al. (2007)	A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers	To detect and classify intrusions from normal behaviors based on the attack type in a computer network
		Abadeh et al. (2007)	Intrusion detection using a fuzzy genetics-based learning algorithm	To describe usage of fuzzy genetics based detect intrusion in a computer network
		Khan et al. (2012)	Application of fuzzy inference systems to detection of faults	Modeling of WIDS
	Machine learning (ML)	Qiming et al. (2000)	Using reinforcement learning for pro-active network fault management	Developing of IDS
		Xu et al. (2005)	A Reinforcement Learning Approach for HIDS Using Sequences of System Calls	Prediction intrusion behavior
		Xu et al. (2007)	Defending DDoS Attacks Using Hidden Markov Models and Cooperative RL	Developing efficient intrusion detection and reaction systems
		Xu et al. (2010)	Sequential anomaly detection based on temporal-difference learning	Anomaly detection- designing method
		Andersen et al. (2009)	Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES)	To detect anomalous activity-analysis component of NIDES to develop baseline profiles of applications
		Agah et al.	Intrusion detection in	Finding the most vulnerable node in

		(2004)	sensor networks: a non-cooperative game approach	a sensor network and protecting it
		Ye et al. (2000)	A Markov chain model of temporal behavior for anomaly detection	Analyzing and detection anomaly behavior
		Devarakonda et al. (2012)	Integrated Bayes Network and Hidden Markov Model for Host based IDS	Prediction intrusion behavior
Multi agent based CI (MCI)	Multi agent system: (MAS)	Renjit et al. (2011)	Multi-Agent-Based Anomaly Intrusion Detection	Development of IDPS
		Fisch et al. (2012)	Learning from others: Exchange of classification rules in intelligent distributed systems	Proposes techniques for IDPS
		Herrero et al. (2009)	MOVIH-IDS: A mobile-visualization hybrid intrusion detection system	Development of IDS
		León et al. (2011)	Towards a Cooperative Intrusion Detection System for Cognitive Radio Networks	Analyzing threats and propose intrusion detection modules
		Vakili et al. (2011)	Coordination of cooperation policies in a peer-to-peer system using swarm-based RL	Devised a self-organized coordination mechanism for cooperation policy setting of rational peers
		Mosqueira-Rey et al. (2007)	A Misuse Detection Agent for Intrusion Detection in a Multi-agent Architecture	Designing a misuse detection agent
		Ramachandra n et al. (2008)	FORK: A novel two-pronged strategy for an agent-based intrusion detection scheme in ad-hoc networks	Designing Anomaly detection algorithm in ad-hoc networks
		Dasgupta et al. (2005)	CIDS: An agent-based intrusion detection system	Designing administrative tool for intrusion detection
		Z. Zhang et al. (2001)	HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification	Design NIDS
		Agah et al. (2007)	Preventing DoS attack in sensor networks: a game theoretic approach	Developing efficient intrusion detection and reaction systems
Critical review		Wu et al. (2010)	The use of computational intelligence in intrusion detection systems: A review	Review
		Kolias et al. (2011)	Swarm intelligence in intrusion detection: A survey	Systematic survey
		Garcia-Teodoro et al. (2009)	Anomaly-based network intrusion detection: Techniques, systems and challenges	Critical Review

	Tsai et al. (2009)	Intrusion detection by machine learning: A review	Review
	Davis et al. (2011)	Data pre-processing for anomaly based network intrusion detection: A review	Review
	Patel et al. (2013)	An Intrusion Detection and Prevention System in Cloud Computing: A Systematic Review	Systematic review

Through this thesis, the trends of TAI, CI, and MCI used in IDPS are studied. Analysis is based on two key aspects concerning the evaluation and comparison of the alternative IIDPS approaches' performances: i.e., the efficiency of the detection process and false alarm rates (i.e. false positives and negatives). The significance of the performance, and especially at this point, the efficiency aspect must be emphasized. As an example, Potyrailo et al. (2012) researched about the influence of wireless chemical sensors based on Radio Frequency Identification (RFID) in the high detection of chemical agents. They claim that any failure in real time diagnosis may lead to harmful events. Wang et al. (2012) achieved a false positive ratio of less than 10% with small packet of buffers to identify the compromised node in wireless networks.

2.1.1 Data Set

Due to the extraordinary hazard of practical operational networks and systems of real environments, performing real time testing is very difficult and complicated. Therefore, most researchers validate the ideas by testing in experimental simulated environments depicting the real environment. There are many datasets that can be used for the detection of abnormal behaviour. For example, in the KDD'99 dataset, certain possible problems were likely to occur, such as, an enormous number of duplicate records have been detected (KDD, 1999).

To examine the possibility of dropped packets by traffic collectors (i.e. TCP dumping) during heavy traffic, there exists different datasets such as NSL-KDD (2009)

which were selected to mitigate the difficulties incurred by KDD'99 datasets. NSL-KDD is significant in that it contains fewer redundant, duplicate records in the training and test phases of learning-based detection. In this manner, the evaluation process of the learning system will not have to be dependent on frequent records. Table 2.2 shows the classification of the datasets based on network traffic.

Table 2.2: Classification of the Datasets for NIDS and WIDS

Name of dataset	Type of dataset	Description of Application domains
KDD'99 (KDD, 1999)	Network IDS	This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.
NSL-KDD (2009)	Network IDS	NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set.
Intel Berkeley Research lab ("Intel Berkeley Research lab," 2004)	Wireless Sensors IDS	Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform.
CRAWDAD data sets (dataset, 2006)	1. Sensor network dataset for enhancing CSMA MAC protocol. 2.Syslog, SNMP, and tcpdump data 3. Dataset of sensor data collected by the CenceMe system.	1. This dataset contains packet transmission traces collected from an experimental wireless sensor network testbed, where E (Enhanced)-CSMA MAC protocol is implemented using TinyOS on Mica2 motes. 2. This dataset includes syslog, SNMP, and tcpdump data for 5 years or more, for over 450 access points and several thousand users at Dartmouth College. 3. CenceMe uses the output of the phones' sensors and external data (if such is available) to infer human presence and activity information. This dataset contains movements and inferred activities of participants using CenceMe on their mobile phones.

According to them, most ML-based NIDSs employ a base line (i.e. a KDD data set and a SVM classifier) for detection of algorithm comparison. Davis et al. (2011) provided a table-based review of the traffic patterns and pre-processing methods utilized by anomaly-based NIDSs. (Patel et al., 2013) presented a comprehensive taxonomy of IDPS into cloud computing, and stipulated a list of requirements for a cloud-based intrusion detection and prevention system (CIDPS). Autonomic computing, ontology, risk management and fuzzy theory form an ideal design to meet the requirements. None, however, have listed or compared the detection performance of Multi agent-based CI (MCI) methods based on IDPS.

2.2 State-of-the-art TAI, CI and MCI in IDPS

Countless studies have suggested means of increasing performance without affecting IDPS quality. Artificial Intelligence (AI) techniques serve to automate the intrusion detection process and reduce human intervention. The process of detecting intrusion is founded on the AI technique by applying methods such as the traditional artificial intelligence (TAI) and computational intelligence (CI) techniques that operate as classifiers. This thesis presents, in detail, the state-of-the-art of TAI, CI and Multi agent-based CI (MCI) systems in the field of IDPS, and additionally highlights the vital concerns/drawbacks to be addressed.

2.2.1 Traditional Artificial Intelligence (TAI)

In TAI techniques, network traffic activity is captured by single classifiers (i.e. fuzzy set, neural network, genetic algorithm and artificial immune system), thereafter, a profile representing its desired behaviour is coded and finally a behaviour model is created. Network events take place, the current profile is assigned and an anomaly score is computed by comparing the two behaviours. The score normally indicates the degree of irregularity for a specific event, such that the IDS raises a flag in the event an anomaly occurs when the score surpasses a certain threshold.

Fuzzy set-oriented IDPSs correspond to an audit data related to a set of rules which identifies different attributes from the training data as a fuzzy rule base (FRB) (Dickerson et al., 2001). FRB is beneficial in instances of misuse but is impractical when dealing with unknown behaviour. To alleviate the drawbacks of unknown behaviour detection, hybrid fuzzy classifiers that consider dynamic fuzzy rule tuning were recommended for a later stage to augment detection rate by dynamically adjusting the rules ((Bridges et al., 2000), (Liang et al., 2005), (Abraham et al., 2007), (Lin et al., 2008), and (Tong et al., 2009)).

The most remarkable advantages of hybrid fuzzy classifier-based IDPSs are robustness and flexibility. Among the most noticeable, disadvantages is the challenge of using a fuzzy set in large scale wireless computing (i.e. excessive resource consumption) and the additional complications imposed on alarm correlation. To moderate the alarm correlation, Kaptanova et al. (2012) suggested a hybrid fuzzy classifier that monitors a temperature value. The fuzzy logic controller performs robust detection, but consumes vast computing resources when performing fuzzy alarm correlation. Thus, to avoid the pronounced resource consumption, a neural network may adopt fuzzy rules by updating the weight of the neurons.

With the intention of minimizing the misclassification of error function, the MLP, SOM and SVM neural networks were implemented to the anomaly-based IDPSs. Neural networks are prominently characterized by their flexibility and adaptability to generate fuzzy rules through performing weight tuning to represent the effective hidden units. This detection approach is frequently employed to detect individual possible misuse (Cannady, 1998; Debar, 1992), to determine which network traffic data clusters contain attacks (Alan Bivens, 2002), to identify deviations from normal behaviour Bankovic et al. (2011), and to judge whether a network visit is normal or not (Y. Li et al., 2012).

A common negative characteristic shared by the proposed variants, from multi-layer feed forward neural networks to self-organizing maps and supporting vector machines (Fisch et al., 2012), is that no expositive structure is provided explaining why a particular detection decision has been made.

A genetic algorithm permits a population of many individuals to infer under distinguished selection rules to a state that maximizes the “fitness function” (i.e. minimizes the cost function), by evolving its operators such as selection, cross over, and mutation. GA contributes another type of anomaly-based IDPS, which is adept at

utilizing communication energy (Khanna et al., 2009) and applying a grammatical evolution (GE) technique with BNF grammar to identify route disruption attacks in mobile ad-hoc networks (Phillips et al., 2010).

The main advantage of this subtype of TAI-oriented IDPS is the capability of a flexible and robust global search method that converges to a global minimum (i.e. a solution from multiple directions), with no prior knowledge of the system's behaviour. Its main disadvantage is the enormous resource consumption involved. If the population is large then the mutation is too great and the system never converges towards a suitable solution. Due to this iteration, an immense amount of resources are consumed.

The observed theory and functions of AIS immunization were inspired by the natural immune system principles; this system's models were then employed in a wide and intricate range of subjects. Clonal selection and negative selection comprise essential shares of this system. The clonal and negative selection functions have a large influence on the security of wireless networks (Hofmeyr et al., 2000).

AIS techniques have been extensively used in negative and clonal selection. They are normally applied as mobile memory detectors to generate several diverse detectors by approximation to achieve lightweight NIDS (Jungwon et al., 2001). Ma et al. (2007) organized WSN as a body, adversaries as pathogens and multi agents as lymphocytes that defend against attacks. In all cases, self-adaptability is facilitated by upgrading the agents' characteristics via the creation of new antibodies. Energy efficiency is attained by deploying decision agents in the base station with sufficient resources and strong computational skills.

AIS functions provide the configurability of driving the gene library evolution by using the clone selection. Also, AIS requires both additional memory and time when

being transferred to generate a vast detector set number. The main disadvantage is its high resource consumption.

2.2.2 Computational Intelligence (CI)

Computational intelligence (CI) classifiers rely on a soft computing (SC) or machine learning (ML) model that allows for the patterns analysed to be categorized. A distinct characteristic of these schemes is the prerequisite for labelled data to train the behavioural model, a procedure that places severe demands on the resources. ML based on SC classifiers is meant to create an iterative process of observing patterns, adjusting to the mathematical form, and making predictions (Alpadin, 2010).

SC classifiers were designed to modify the classification performance of TAI methods by incorporating a multifold learning algorithm (Zadeh, 1994). SC classifiers distinguish two main approaches: Neuro Fuzzy (NF) and Genetic Fuzzy (GF) models. NF and GF are a combination of a fuzzy set with NN and GA, which are utilized to adjust the structure and parameters of a fuzzy system by neural network and genetic algorithm operators, respectively. The goal is an optimal, continuous membership function that identifies anomalous behaviour with supervised monitoring abilities, high detection rate and low false alarm rate (Buckley et al., 1994; Fullér, 2000).

Neuro fuzzy is a combination of a fuzzy set and adaptive neural network that tunes the fuzzy membership function using neural networks. Neuro-fuzzy techniques are found in the milieu of NIDS, generally applied to IDS problems (Gomez et al., 2002). The multi-layer perceptron learns the fuzzy rule, after which this neural network performs a fuzzy interface process to identify attacks (Chavan et al., 2004; Mohajerani et al., 2003).

In some ways associated to anomaly-based IDS methods, a revised Neuro-fuzzy classifier with a GA was proposed to modify the fuzzy engine for detection (Toosi et al., 2007). At the same time, IDS employs the fuzzy genetic learning method to construct a primary population by using the fuzzy rule. The antecedent fuzzy part provides a uniform crossover for a pair of fuzzy rules, following which the antecedent fuzzy set randomly supersedes the fuzzy set with a mutation probability. Lastly, the fuzzy genetic method terminates the fuzzy classifier execution through a total number of generations (Abadeh et al., 2007). More recently, Khan et al. (2012) developed a fault detection strategy in WSN. In this system, a Recurrent Takagi-Sugeno-Kang FIS (RFIS) strategy decided whether or not to declare the node malicious.

In all SC techniques fuzzy logic is optimized to enhance the detection accuracy. The number of false positives is reduced and only the true positive intrusion events from the raw audit data are increased. It is still challenging though to tune the fuzzy rules based on IDS into WSN to lessen the false positives and boost detection rate.

In the expansion of IDPS, the ultimate aim is to obtain a high level of accuracy in the various intruder detection schemes. Several ML-based designs have been applied to IDPS. Some of the most important ones are cited below, and their main strengths and weaknesses are identified.

Reinforcement learning (RL) appears to be a greatly significant method of wireless network security due to its capability to autonomously learn new attacks via online, unsupervised learning, as well as modify new policies without complex mathematical approaches (Barto., 1998). RL has been proven to be effective, especially in real time detection and when no prior system behaviour information is assumed.

RL constitutes another form of computational intelligence-based techniques, capable of forecasting online network fault detection by Partly Observable Markov

Decision Procedure (POMDP), a practice which transforms a reward function into Markov chains (Li et al., 2014). Hence, to execute the learning prediction, a TD learning algorithm was employed. A value function forecasting model was constructed upon completion of the learning phase.

In a non-cooperative game theory model, the system uses a Q-learning algorithm for any adversary recognition in sensor networks (A. Agah et al., 2004). Temporal difference sequential anomaly detection (TD-SAD) aided by the Markov reward model is used to determine data labelling and improve its detection model performance (X. Xu, 2010). The multilayer RL framework assisted by HMM was proposed to solve real-time detection in a complex state space (Andersen et al., 2009).

The primary disadvantage of reinforcement learning is the abundance of resources consumed, in other words, the lack of memory to sustain the agent's data. The agent's memory is stored in a look-up table, or the Q-table. The values fill the Q-table with the maximum positive rewards possible when executing an action from the current state to the next-state space. Consequently, the high values in the Q-table expend all these resources. RL, when compared with other soft computing methods (i.e. neuro fuzzy and genetic fuzzy), is highly dependent on state space.

The Bayes Network principle provides a distribution possibility to encode statistical relationships among any single quantity. BN is based on Bayes' theorem (alternatively, Bayes' law), which imparts a means to apply quantitative reasoning. This model is normally used for IDS in combination with HMM and MDP, a procedure that yields several advantages (Ahmadabadi et al., 2001). Anomaly detection has the ability to represent a norm profile of temporal behaviour by shifting the observation window to view the last N audit event continuously and detect intrusion behaviour during window observation (Ye, 2000).

BN-based HMM techniques have been utilized in IDS, typically applied to compute the statistical feature of normal behaviours incoming from the IP source (Xin Xu et al., 2007). In a dynamic-based IDS, BN and HMM were recommended to identify intrusion. They reduced the KDD data set by choosing five attribute numbers for the training mode in TCP connection. Then, the dynamic Bayesian network was initialized with the Baum-Welch algorithm to classify normal and attack data (Devarakonda et al., 2012).

As pointed out (Patcha et al., 2007), a serious disadvantage of using BNs is that the accuracy of this method is dependent on certain assumptions characteristically based on the behavioural model of the target system; deviating from these assumptions decreases accuracy. Selecting an accurate model is the first step towards solving the problem, as a result, considerably higher computational effort is required.

In most ML techniques, reinforcement learning (RL) utilizes HMMs and BNs to improve the detection operation. The main effort therefore goes to optimizing RL through the Fuzzy Logic Controller (FLC).

2.2.3 Multi Agent-based Computational Intelligence (MCI)

MCI techniques function by applying the multi agent system (MAS) to computational intelligence (CI) in order to enhance the performance of detection and response. On the other hand, cooperative MAS uses CI methods such as SOM, SVM, GA, RL and Game Theory to determine temporal behaviour and respond to any deviation. The main objective of MCI consists of distributing MAS to each cluster to provide a CI mechanism that makes individual and cooperative decisions associated to IDPS (Wooldridge, 2009). MCI has been widely employed in the domain of network security, especially in WIDS (W. Li et al., 2012) and cloud computing-based IDS (Doelitzscher et al., 2012).

MCI approaches correspond to hierarchical multi-agent architecture for intrusion detection, and modify statistical models like SRI's NIDES statistical algorithm. The advantage is access to a distributed, three-layer intrusion detection. As such, each module negotiates with other agents from the lower to higher tiers, seeking to overcome detection complexity (Z. Zhang et al., 2001).

With regards to cognitive sensors, a model was proposed in two phases, namely, which local agents use support vector machines in the training mode and which local agents use mobile agents in the decision mode to classify suspicious behaviour (León et al., 2011). Some methods suggest a multi-agent system where each local agent collects data through a mobile agent. The local agent then examines the integrity of the system by a SVM classifier at the time an attacker enters the system. Also, in the communication mode the mobile agent verifies activity; if there is no suspicious activity, the message is forwarded to a neighbouring node. The decision making component of detection is based on the Bayes theory, in which, if the probability of normal activity is smaller than the assumed abnormality threshold, the current activity is categorized as abnormal (Renjit et al., 2011).

MCI-based algorithms are intended to classify audit data according to a set of fuzzy associated rules. First, a Java agent-based snort collects packets with a packet sniffer and then creates an input data for the rule engine. Subsequently, the rule engine forwards the pattern matching algorithm to a multi-agent system. The audit data is then classified accordingly (Mosqueira-Rey et al., 2007). Intrusion detection utilizes MAS along with the Fuzzy Classifier System (FCS) and Knowledge Base to detect abnormal activities (Dasgupta et al., 2005).

With respect to the communication mechanisms, Vakili et al. (2011) developed a cooperation policy setting process. Interacting peers' agents regard each other's

reliability or reputation as an impact factor influencing the value of information received from the other, and hence their learning mechanisms. A broadcasting mechanism was proposed for knowledge acquisition in dynamic environments based on probabilistic modelling which was improved through other cooperative communication (Fisch et al., 2012).

In some domains of WSN-based IDPS, a two-tier, MAS-based Ad-hoc intrusion detection mechanism was proposed. The first tier runs an auctioning-based mechanism for node task allocation. The second tier consists of the classification algorithm that uses a variation of an Ant Colony Optimization to identify the anomaly level (Ramachandran et al., 2008). The collaborative work of the tactical squad of agents has led to concurrent detections of multiple sinkhole threats from different routes in the ad-hoc network (Stafrace et al., 2010). Specifications of collaborative IDPS were developed by employing MAS characteristics. For instance, the inclusion of deliberative (CBR-BDI) agents -- a combination of CBR agents' life cycles (i.e. retrieval, reuse, revision and retention stages) seems appropriate for packet-based detection (Herrero et al., 2009). For this purpose, a CBR life cycle with a cooperative version of Maximum Likelihood Hebbian Learning (MLHL) is reflected upon.

Multi agent-based IDPS, in terms of CI (MCI) and non-CI methods, have emerged in commercial products. In recent years, a number of pioneering systems from MCI-based IDPS, i.e. C-Sniper System, have been practically adopted by US forces. Such a system automatically detects and neutralizes enemy snipers (DARPA, 2012). In brief, this thesis attempts to highlight the possible beneficial impact of MCIs using Fuzzy Logic and Reinforcement Learning, as well as to point out potential pitfalls of not integrating MCI into Co-IDPS. Figure 2.1 indicates a chronological list of TAI, CI and MCI-based IDPS events with respect to the relevant technologies.

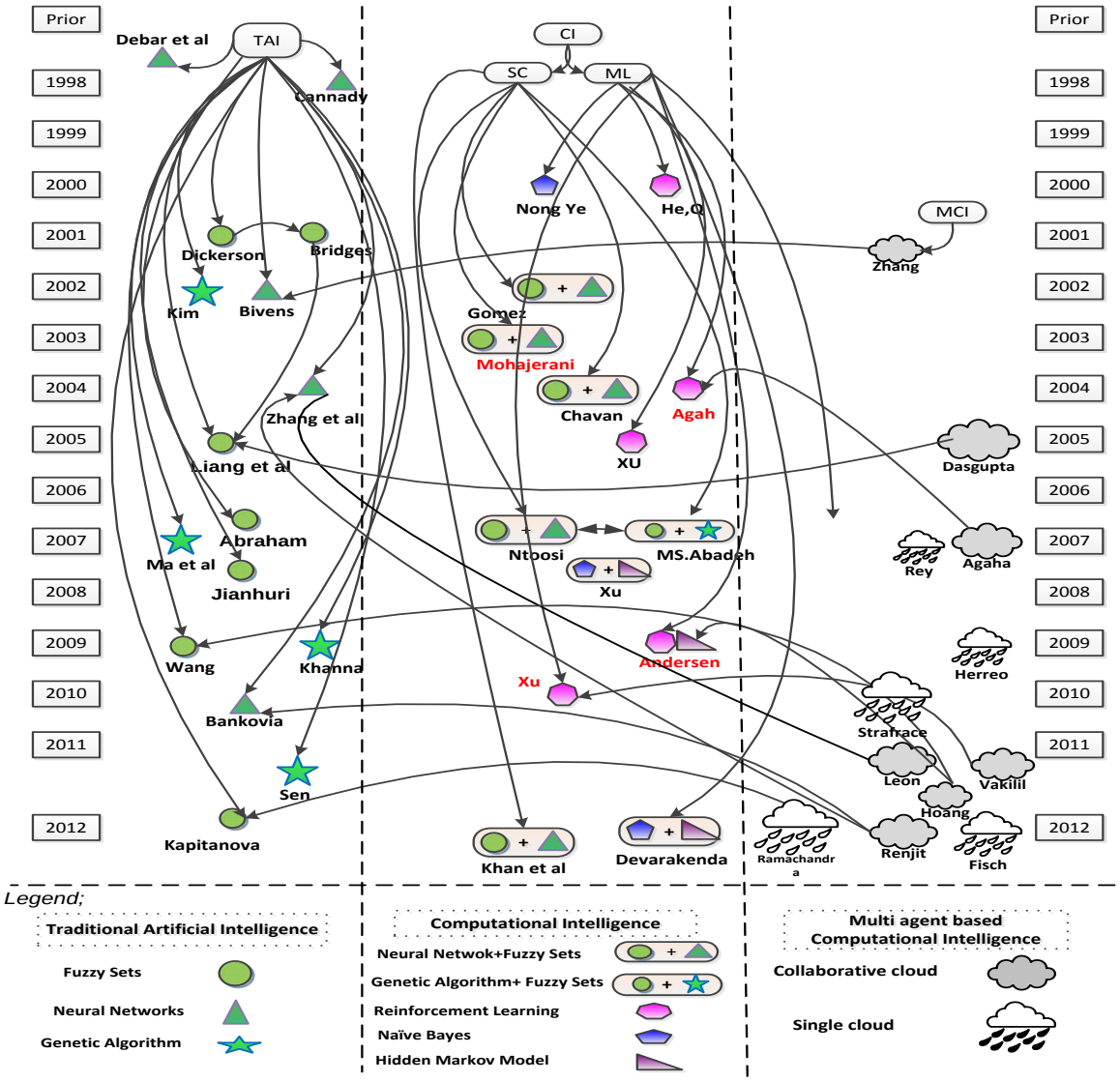

Figure 2.1: Chronological order of [TAI], [CI] and [MCI] based IIDPS

Figure 2.1 illustrates the chronology of those artificial intelligent techniques which focus on TAI (i.e. fuzzy set, neural network and evolutionary algorithm), CI (i.e. soft computing and machine learning) and MCI. The figure summarizes the MCI techniques into two clouds: collaborative and single clouds. A collaborative cloud is a Multi Agent System (MAS) making use of TAI methods and in most cases comprises CI approaches. For example, reinforcement methods utilize neural networks and a fuzzy set (Renjit et al., 2011). A single cloud is a multi-agent-based detection mode that does not use CI and TAI. As proposed in (Ramachandran et al., 2008), a MAS follows an auction and reputation mechanism for performing the task allocation in intrusion

detection. The two cloud types were compared with the TAI and CI methods. The summary implicates that detection and the response system based on the two MAS-based clouds perform extremely well.

2.3 Detection rates performance method and criteria for evaluation

The effectiveness of an IIDPS is assessed on how capable the detection method is to make correct predictions. According to the real nature of a given event compared to an IIDPS prediction, four possible outcomes are shown in Table 2.3. The outcomes are known as the IIDPS reaction matrix. True negatives (TN) as well as true positives (TP) correspond to a correct IDS operation; that is, events are successfully labelled as normal and attack, respectively. False positives (FP) refer to normal events predicted as attacks, while false negatives (FN) are attacks incorrectly predicted as normal events ("Intel Berkeley Research lab," 2004).

Table 2.3: Possible status for an IIDPS reaction

		Predicted	
		Normal	Attack
Actual	Normal	True Negative(TN)	False Negative(FN)
	Attack	False Positive(FP)	True Positive(TP)

A high FP rate that seriously affects the system's performance can be detected, and an elevated FN rate leaves the system vulnerable to intrusions. Both FP and FN rates ought to be minimized, together with maximizing TP and TN rates. Based on Eqs. (2.1) to (2.6) and the IIDPS reaction matrix, a possible status for an IIDPS reaction is shown. It applies the following measures to quantify IDS performance (Blasco et al., 2010):

$$\text{True negative rate (TNR)} = \frac{\mathbf{TN}}{\mathbf{TN+FP}} = \frac{\mathbf{no.\ true\ alerts}}{\mathbf{no.alerts}} \quad (2.1)$$

$$\text{True positive rate (TPR) or Sensitivity or Recall (R)} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{\mathrm{no.\ detected\ attacks}}{\mathrm{no.\ observables\ attack}} \quad (2.2)$$

$$\text{False positive rate (FPR)}: \frac{\mathrm{FP}}{\mathrm{TN+FP}} = 1 - \frac{\mathrm{TN}}{\mathrm{TN+FP}} \quad (2.3)$$

False negative rate (FNR): $\frac{\mathrm{FN}}{\mathrm{TP+FN}}$ (2.4)

Accuracy= $\frac{\mathrm{TN+TP}}{\mathrm{TN+TP+FN+FP}}$ (2.5)

Precision= $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$ (2.6)

Most of the systems employed in the current research works used the same evaluation metrics such as the Detection Rate and False Alarm Rate. Some researchers have addressed the problems of IDS by proposing new DR and FAR. Table 2.4 shows the proposed evaluation metrics by researchers.

Table 2.4: Evaluation metrics proposed by authors

Authors, Paper	Accuracy of Intrusion	False alarm rate	Description
Intrusion detection through learning behaviour (Balajinath et al., 2001)	$\text{Accuracy} = \left[1 - \frac{\mathrm{n}}{\mathrm{N}}\right] 100, \text{FAR} = \left[\frac{\mathrm{n}}{\mathrm{N}}\right] 100$		Where n is the count of command samples that are in total command set S after current command, N the initial size of total command set.
Design and performance evaluation of a lightweight wireless early warning intrusion detection prototype. (Fragkiadakis et al., 2012)	Score=b*(c−d)		Where $d=\sqrt{\mathrm{FAR}^2 + (1-\mathrm{DP})^2}$ is the distance of a trade-off point (for a specific threshold h) from the optimum point (DP = 1 and FAR = 0), and b,c ∈ R+.
Optimization of load balancing using fuzzy Q-Learning for next generation wireless networks (Muñoz et al., 2013)	$\mathrm{U} = [\mathrm{CBR} + (1 - \mathrm{CBR}) \cdot \mathrm{CDR}] \cdot .100$		Where U is a metric that aggregates both key performance indicators to provide an estimation of the user dissatisfaction. Such indicators, CBR and CDR, consider the total number of blocked and dropped calls in the network, respectively.
Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining (Huang et al., 2013)	$\mathrm{U} = \rho * \mathrm{SP} - \beta * \mathrm{FN} - \theta * \mathrm{FP}$		Where U is a utility, SP Represents true positive rate of attack patterns. There are attacks and defences, FN Represents false negative of attack patterns, FP Represents false positive of attack patterns, ρ Represents the weight of successful prediction, β Represents the weight of failed prediction, and θ Represents the weight of failed prediction.
Measuring Intrusion Detection Capability: An Information-Theoretic Approach (Gu et al., 2006)	$C_{ID} = \frac{I(X;Y)}{H(X)}$		Let X be the random variable representing the IDS input and Y the random variable representing the IDS output.

Traditionally, intrusion detection and prevention approaches are studied from two major views, namely anomaly and misuse detection though no considerable difference in characteristics exists between them. Liao et al. (Huang et al., 2013)proposed a subdivision of detection approaches into five subcategories, including statistics-based, pattern-based, rule-based, state-based and heuristic concepts, but the properties of intelligent detection approaches are not defined. Due to the lack of a more detailed view of detection and prevention approaches using multi agent system-based computational intelligence, this thesis presents a classification of three subclasses with an in-depth perspective on the characteristics: traditional artificial intelligence-based, computational intelligence-based, and multi agent-based CI. Accordingly, we have carefully assembled the current intrusion detection approaches, especially those found in wireless networks (Table 2.5).

Table 2.5: Classifications and comparisons of various intrusion detection approaches

	Detection approach			**methodology** [a]		**Technology type** [b]	**Detection of attack** [c]	**performance** [d]	**Type of source** [e]	**characteristics**
				AD	MD	H/N/W	K/U/B	H/M/L	P1/P2/N	Flexibility and adaptability
Traditional Artificial Intelligence	Neural network	No. of articles:(6)	Debar et al. (1992)		√	Host-based (H)	Known attacks (K)	Low (L)	Public dataset(P1)	
			Cannady et al. (1998)		√	Network-based (N)	Known attacks (K)	Low (L)	Public dataset(P1)	
			Zhang et al. (2000)	√		Wireless -based (W)	Known attacks (K)	Moderate (M)	Private dataset(P2)	
			Bivens et al. (2002)	√		Network-based (N)	Known attacks (K)	Moderate (M)	Private dataset(P2)	
			Bankovic et al. (2011)	√		Wireless -based (W)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	
			(Yan Li, 2012)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	
	Fuzzy Sets	No. of articles:(6)	(Dickerson et al., 2001)		√	Network-based (N)	Known attacks (K)	Low (L)	Public dataset(P1)	Robustness and flexibilities
			Bridges et al. (2000)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Private dataset(P2)	
			(Liang et al., 2005)	√		Wireless -based (W)	Unknown attacks (U)	Moderate (M)	Private dataset(P2)	
			(Abraham et al., 2007)	√		Network-based (N)	Both known and unknown attacks (B)	Moderate (M)	Private dataset(P2)	
			(Jianhui et al., 2008)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Private dataset(P2)	
			(Tong et al., 2009)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	Private dataset(P2)	
	Artificial Immune system	Articles(2)	(Jungwon et al., 2001)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	Flexible and robust in global search methods
			(Ma et al., 2007)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	Public dataset(P1)	

	Genetic algorithm	Articles(2)	(Gu et al., 2006)	√		Wireless -based (W)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	Flexible and robust in global search methods
			(Sevil Sen, 2011)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	Public dataset(P1)	
	Total articles: (16)									
Computational Intelligence	Soft computing (SC)	No .articles(6)	(Mohajerani et al., 2003)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	Lower false positive rate, high accuracy
			(Gomez et al., 2002)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	
			(Chavan et al., 2004)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	Public dataset(P1)	
			(Toosi et al., 2007)	√		Network-based (N)	Unknown attacks (U)	High (H)	Public dataset(P1)	
			(Abadeh et al., 2007)	√		Network-based (N)	Unknown attacks (U)	High (H)	Public dataset(P1)	
			(Khan et al., 2012)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	Public dataset(P1)	
	Machine learning	No articles(8)	(Qiming et al., 2000)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	unspecified dataset(N)	High accuracy, Self-learning, Fault tolerant
			(Xin Xu et al., 2005)	√		Host-based (H)	Unknown attacks (U)	Moderate (M)	unspecified dataset(N)	
			(Xin Xu et al., 2007)	√		Network-based (N)	Known attacks (K)	Moderate (M)	unspecified dataset(N)	
			(X. Xu, 2010)	√		Network-based (N)	Unknown attacks (U)	Moderate (M)	unspecified dataset(N)	
			(Andersen et al., 2009)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(A. Agah et al., 2004)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			Ye et al. (Ye, 2000)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Devarakonda et al., 2012)	√			Unknown attacks (U)	Moderate (M)	unspecified dataset(N)	
	Total articles: (14)									

Multi agent based CI(MCI)	Multi agent system used CI (MAS)	No .articles(10)	(Renjit et al., 2011)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	Distributed, high overall security, cooperative
			(Fisch et al., 2012)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Herrero et al., 2009)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(León et al., 2011)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Vakili et al., 2011)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Mosqueira-Rey et al., 2007)		√	Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Ramachandran et al., 2008)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Dasgupta et al., 2005)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Z. Zhang et al., 2001)	√		Network-based (N)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
			(Afrand Agah et al., 2007)	√		Wireless -based (W)	Unknown attacks (U)	High (H)	unspecified dataset(N)	
	Total articles: (10)									

a. Detection methodology: anomaly-based detection (AD), misuse-base detection (MD)

b. Technology type: host-based (H), network-based (N), wireless -based (W)

c. Detection of attacks: known attacks (K), unknown attacks (U), both known and unknown attacks (B)

d. Performance: high (H), moderate (M), low (L)

e. Type of source: Public dataset(P1), Private dataset(P2), unspecified dataset(N)

Table 2.5 indicates the types and subtypes of intrusion detection. The detection methodology is categorized into anomaly and misuse-based. It contains a host-based, network-based and wireless -based intrusion detection and prevention systems. The type of attack detection is classified into known, unknown, and both kinds of detection. The performance indicates detection efficiency, while the level of performance is evaluated by degrees of high, moderate and low. The sources comprise of a public dataset (e.g. KDD99), a private dataset (e.g. NSL-KDD) and an unspecified dataset, as extracted from previous attacks. The available data is utilized to differentiate intrusion behaviour from suspicious activities. The methodology of anomaly-based MAS and CI compared with TAI satisfies the detection, particularly in unknown attacks. Detection efficiency in the multi agent-based computational intelligence (MCI) method portrays superior performance. The most significant aspects of the MCI-based IDPS mentioned are high accuracy, self-learning, and robustness.

Figure 2.2 depicts the number of manuscripts investigated over a 14 year period from 1998 to 2012. The amount of manuscripts regarding TAI detection methods reached a peak in 2002, declined gradually by 2006, then remained stable until now. It is not easy to apply those TAIs which mitigate IIDPS vulnerabilities, thus a new generation of intelligent attacks can arise. Nevertheless, CI and MCI have received increasing consideration recently.

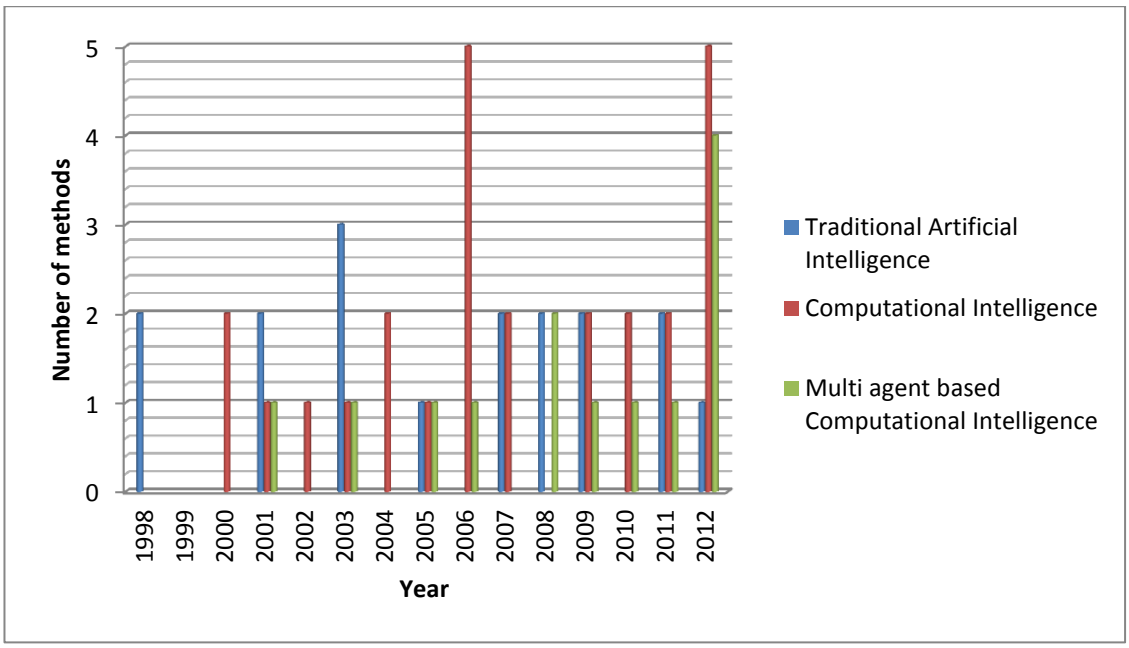

Figure 2.2: Year wise distribution of articles for the various types of classifier layouts

From Figure 2.2 it can be observed that while traditional artificial intelligence, computational intelligence and multi agent based computational intelligence were fairly static from 2007 to 2009, however, CI has become more popular than TAI by the year 2012 because CI in terms of MCI shows better performance of intrusion detection and false alarm rate.

2.3.1 Traditional Artificial Intelligence

The emergent anomaly detection applications have brought about a new trend of IDPS-focused research which concentrates on ways of managing alarms. Table 2.6 lists the most recent research attempting to deal with intrusion detection and prevention problems based on Traditional Artificial Intelligence approaches.

Table 2.6: Classification of Traditional Artificial Intelligence-based IDPS

Reference	Method	Objective	Performance	Technology category	Type of attacks
Zhang et al. (Z. Zhang et al., 2001)	Combining data mining with fuzzy rule	To identify misused behavior in a network	Up to 50% increase in DR	Hybrid fuzzy(FRB)	Individual
Bridges et al. (2000)	Fuzzy association rule	To learn a normal pattern	Reduced FAR by 20%		Individual
Liang et al. (2005)	A fuzzy Logic approach	To identify the abnormal	Increased the d Detection rate to		

	combined with double sliding window detection	behavior in sensor networks	99.97% and reached to 0.05% FAR		
Abraham et al. (2007)	Fuzzy classifier uses a decision tree	To detect attribute anomalies	Averaged 90% in DR		Public
Lin et al. (2008)	The data mining uses PTBA algorithm to extract rule mining	To classify network traffic behavior	More than 90% increase in Detection rate (DR)		Public
Kapitanova et al. (2012)	Combining rules with similar consequences and removing negligible rules.	To modify the precision of event detection	False positive rate 0% compares with 0.13% decision tree and 1.56% Naïve bayes theory.		
Tong et al. (2009)	Fuzzy C-mean clustering algorithm calculates the distance between connection record all the actual cluster through a non-linear function	To distinguish the normal cluster and abnormal	Detection rate of 96% can be reached, if false positive rate is controlled to less than 1.5%	Fuzzy C-mean(FCM)	Individual
Cannady et al. (1998)	Multilayered feed-forward comprise MLP	To detect misuse instances such as SYNFlood	Increased DR by 60%	Multilayered feed-forward (MFF)	Individual
Debar et al. (1992)	Modular intrusion detection based on neural networks and expert systems	Prediction error rate	Average 30%, increasing DR	Support Vector Machine(SVM)	Individual
Bivens et al. (2002)	SOM was utilized as a clustering method for MLP neural networks	Attack detection.	Up to 98% increase in DR,but with a small reduction of FAR	Self- organizing Map(SOM) and Multi- layer perceptron(MLP)	Public
Bankovic et al. (2011)	Using Euclidean distance throughout the reputation self-organizing map algorithm.	To detect deviations from normal behavior.	More than 80% detection rate	SOM	Individual
Khanna et al. (2009)	Four fitness functions used: 1. Monitoring node integrity fitness (MIF) 2. Battery Fitness (MBF) 3.Coverage fitness (MFC) 4. Cumulative trust fitness (CTF)	To maximize the performance of IDS	Decrease the FAR,but with a small increase in DR(60%)	Genetic Algorithm(GA)	Individual
Sen et al. (2010)	Genetic programming applied a grammatical evolution (GE) technique that uses	To identify the route disruption attacks in mobile ad-hoc networks	More than 99.41% increase in DR and reduced false positive by 1.23%	Genetic Algorithm(GA)	Individual

	BNF grammar				
Jungwon et al. (2001)	The gene library evolution by using clone selection (Co-Evolution)	To identify misuse	No test	Artificial Immune System	
Ma et al. (2007)	Inspired by immune system used WSN as a body, adversary as a pathogens and multi agents as lymphocytes	To defend against attacks	No test	Artificial Immune System	

Generally, TAI techniques for IDPS are categorized into three technologies. The fuzzy set-based WIDSs consist of a fuzzy rule- based (FRB) and Fuzzy C-mean (FCM); the neural network-based IDPS comprises a Multi-layer Perceptron (MLP), Support Vector Machine (SVM) and a Self-Organizing Map (SOM). Finally, the evolutionary algorithm is made of a genetic algorithm (GA) and an artificial immune system (N. Li et al.).

The majority of researchers working with TAI have provided some solution to appraise the performance of IDPS for anomaly techniques (refer to Table 2.3) since anomaly techniques based on hybrid TAI (i.e. using fuzzy data mining) generate more accuracy than single TAI techniques (i.e. self-fuzzy), as per Figure 2.3. The hybrid TAI approach, such as combining the data mining techniques with fuzzy set for instance, optimizes the system's visibility and performance. Moreover, the false alarm correlation and detection rate becomes more complicated. This is why it is necessary to attract researchers' awareness to attempt and provide solutions to IDPS management in the recently utilized hybrid TAI detection methods.

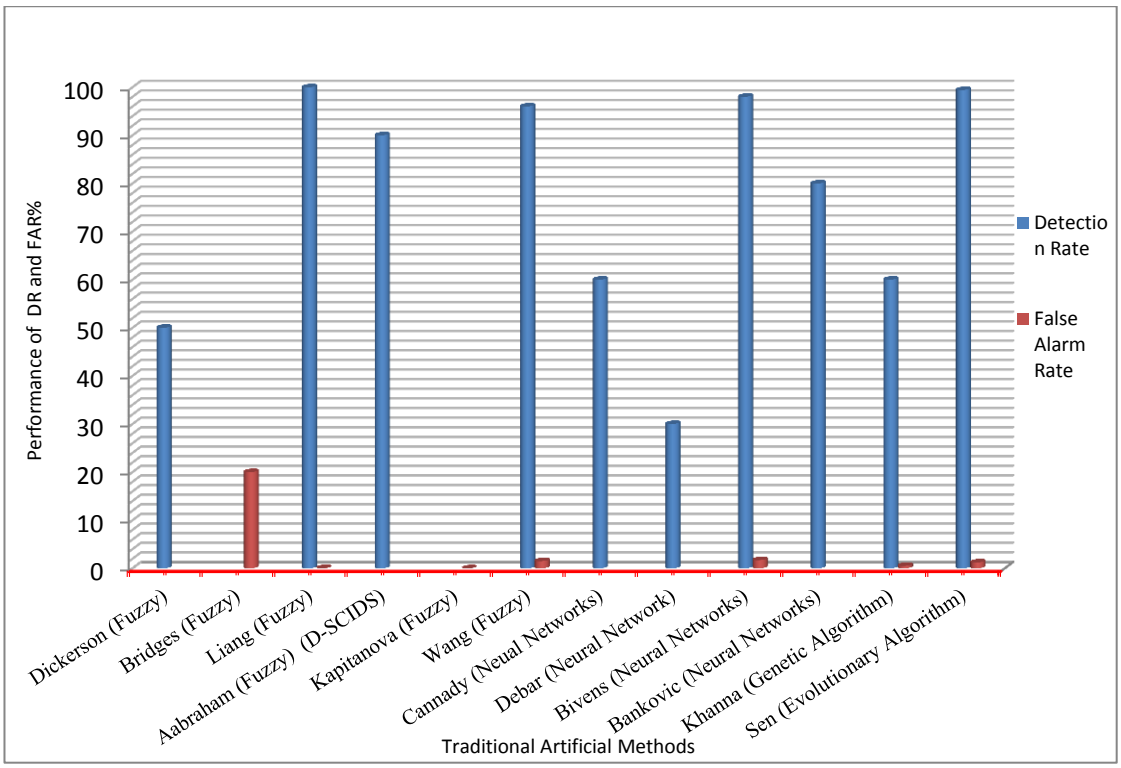

Figure 2.3: Comparison of TAI methods in terms of detection rate and false alarm rate

TAI methodologies imply that SVM and fuzzy rule-based (FRB) are increasing the performance of detection rate (DR) and false alarm rate (FAR) in the network environment. On the contrary, FCM, SOM, MLP, GP and AIS do not consider IDPS due to their inability to provide adaptability at a time when adversary behaviour is changing dramatically.

2.3.2 Computational Intelligence

Two strategies, namely machine learning (ML) and soft computing (SC), are utilized for designing intelligent intrusion detection. The objective of this literature is to introduce SC and ML in terms of Computational Intelligence-based IDPS.

The most recent examined works applicable to CI-based IDPS are ordered in terms of soft computing (SC) and machine learning (ML) taxonomy. The methods applied are very similar to each other. For instance, the neural network-based fuzzy solution replaces fuzzy-based neural network classifier to tune the fuzzy rule to achieve

more accurate detection. The most important features that are different between the CI and TAI methods are prevention capability and response. Currently conducted research and proposed solutions with respect to CI techniques are still a long way from an ideal IDPS. They not only lack the desired collaborative IIDPS characteristics, but also fail to reach into wireless network territory. In all circumstances, CI techniques provide high accuracy, self-learning, and are fault tolerant, but they are not capable of taking into consideration all the features addressed in IDPS such as energy efficiency, detection rate and false alarm rate. These inefficiencies are evidence of the lack of distributed and cooperative knowledge of the proper requirements identified prior to initiating any development.

Table 2.7: Classification of computational intelligence in IDPS

Reference	**Method**	**Objective**	**Performance**	**Technology category**
Mohajerani et al. (2003)	Decision making based on fuzzy and neural networks	Traffic monitoring	Normal pattern detection accuracy, 6% greater than attack detection accuracy. Meanwhile, the false alarm rate is around 9.4%	Neuro fuzzy
Gomez et al. (2002)	Classification process using fuzzy and genetic	Intrusion detection	The FAR reduced to 5% with a correct detection rate of 98.5%	Genetic fuzzy
Chavan et al. (2004)	Using rule based decision tree and neural network for classification	To encounter vulnerabilities present in snort and classify anomaly behavior.	No test	Neuro fuzzy
Toosi et al. (2007)	A revised neuro-fuzzy classifier with a GA	To modify the fuzzy engine for detection	Up to 95% increase in DR,but the incorrect detection rate is 1.9%	Neuro fuzzy
Abadeh et al. (2007)	System provides the crossover and mutation by using fuzzy rules.	Intrusion detection	Up to 99.08% increase in DR, but with a small reduction of false alarm (3.85%)	Fuzzy genetic
Khan et al. (2012)	The initial FIS is trained by using neural network	Intrusion detection	No test	Neuro-fuzzy
Qiming et al. (2000)	Partly observable Marko decision procedure transforms a reward function into Markov chains	To proactive network fault detection.	No test	Reinforcement learning(RL
Xu et al. (2007)	Cooperative RL uses HMM to compute the statistical feature of normal behaviors	DDoS detection	97% correct detection rate obtained with zero FAR	HMM-RL

	incoming from the IP source.			
Xu et al. (2005) Xu et al. (2010)	Temporal difference sequential anomaly detection (TD-SAD) aided by the Markov reward model	To figure out data labeling and improve detection model performance.	More than 98% detection rate and very low false alarm rate	Reinforcement learning and MDP
Andersen et al. (2009)	A multilayer RL framework uses Hidden Markov Model	To actualize the detection of DDoS attacks	No test	Reinforcement learning and HMM
Agah et al. (2004)	A non-zero-sum game theoretic uses Q-learning algorithm to establish the Nash equilibrium.	For adversary recognition in sensor networks.	Average 50% in DR	Reinforcement learning and game theory
Ye et al. (Ye, 2000)	Moving the observation window by using Markov chain model	To represent a norm profile of temporal behavior	Detection rate increased (100%) and false alarm rate decreased (0%).	Markov chain model
Devarakonda et al. (2012)	Dynamic Bayesian network was initialized using the Baum-Welch algorithm to reinforce probability of the partial observation sequence	To identify intrusion	The high-count attack value (0.624) is greater than low-count attack (0.228)	Bayesian network and hidden Markov

The performance evaluations imply that Reinforcement Learning and Neuro Fuzzy are the most perceived in CI classifiers based on IDPS. As shown in Figure 2.4, RL facilitates a high level of accuracy for the detection process. Alternatively, the process of detection may attain superior accuracy upon autonomic agent decision making.

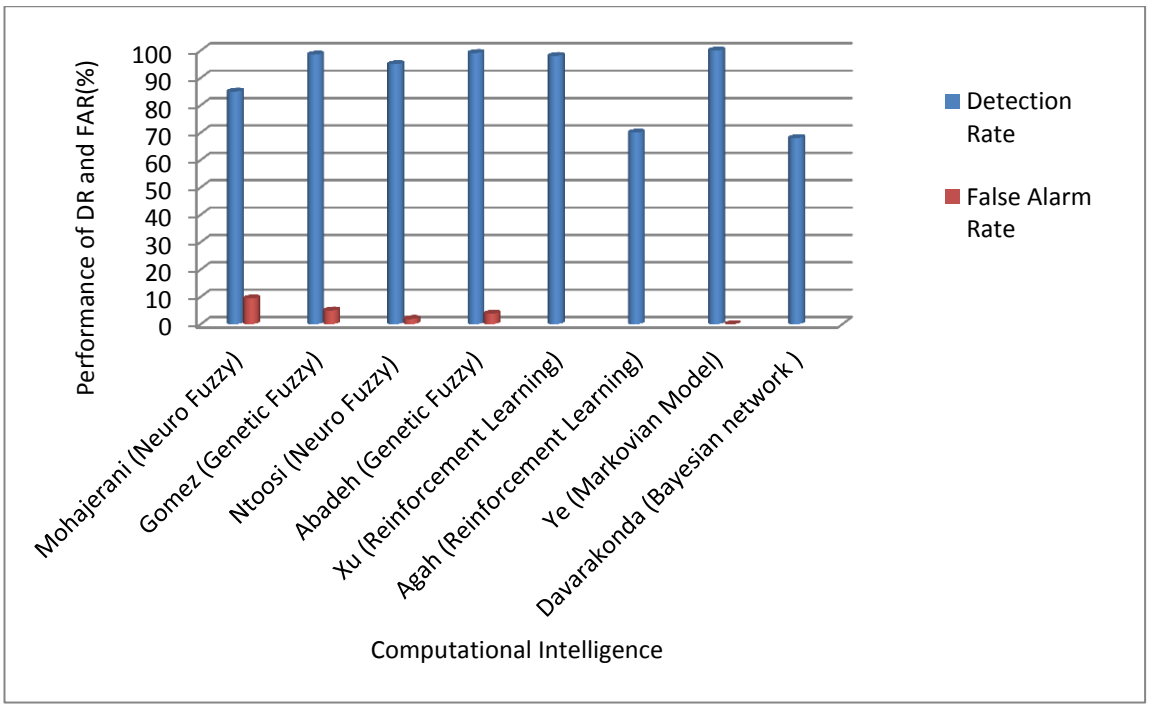

Figure 2.4: Comparison of CI methods in terms of detection rate and false alarm rate

2.3.3 Multi agent system-based Computational Intelligence

In line with the progress made on launching MAS, numerous Intelligent Intrusion Recognition systems apply this sort of cooperative classifier into computational intelligence. Table 2.8 illustrates the percentage of all research articles implementing multi agent system techniques to the CI-based WIDS methods. The results indicate that MASs are becoming increasingly perceived for SC&ML classifier design.

Table 2.8: Classification of multi agent computational intelligence-based IDPS

Reference	Method	Objective	Performance	Technology category
Zhang et al. (2001)	Using distributed three-layer intrusion	NIDS	Average 70% in DR and 9% FAR	MAS-NN
Mohajerani et al. (2003)	Decision making based on fuzzy and neural networks.	Traffic monitoring	Normal pattern detection accuracy, 6% greater than attack detection accuracy. Meanwhile, the false alarm rate is around 9.4%	Neuro fuzzy
Dasgupta et al. (2005)	Data mining uses MAS	To detect abnormal behavior	No test	MAS along with Fuzzy Classifier System (FCS) and Knowledge Base (KB)
Agah et al.	Cooperative game	To detect		

(2007)	theory	attack		
Mosqueria-Rey et al. (2007)	Rule engine uses the pattern matching algorithm	To packet sniffer	No test	MAS-FRB
Ramachandran et al. (2008)	Two tires perform: an auctioning based mechanism for task allocation of nodes. The classification algorithm using a variation of Ant colony optimization	To identify the level of anomalous	Increased the correct attack detection to 79% and reach to 4% false positive	MAS-Swarm Intelligence
Herrero et al. (2009)	Combines CBR life cycle (i.e. retrieval, reuse, revision and retention stage) with a cooperative version of Maximum Likelihood Hebbian Learning (MLHL).	Wireless based IDPS	No test	MAS-probabilistic modeling
Stafrace et al. (2010)	Agent based adhoc-network	To concurrent detections of multiple sinkholes threat	Obtained 86% correct detection rate and 5% false positive	MAS-Collaborative
Renjit et al. (2011)	Using mobile agent	IDPS	No test	MAS-SVM-SOM-BN
Leon et al. (2011)	A Multi agent system utilized supports a vector machine in training mode	To classify suspicious behavior	No test	MAS-SVM
Vakili et al. (2011)	A reputation assignment mechanism in the developed cooperation policy setting process	IDPS	No test	MAS-RL
Fisch et al. (2012)	A broadcasting mechanism communication uses knowledge acquisition	Co-IDPS	No test	MAS-probabilistic modeling
Devarakonda et al. (2012)	Dynamic Bayesian network was initialized using the Baum-Welch algorithm to reinforce probability of the partial observation sequence	To identify intrusion	The high-count attack value (0.624) is greater than low-count attack (0.228)	Bayesian network and hidden Markov

Figure 2.5 demonstrates the correct detection rate achieved by MCI, which steadily increases as the false alarm rate dramatically decreases. Without a doubt, MCI approaches may potentially reach enhanced flexibility, making them even more popular in the near future.

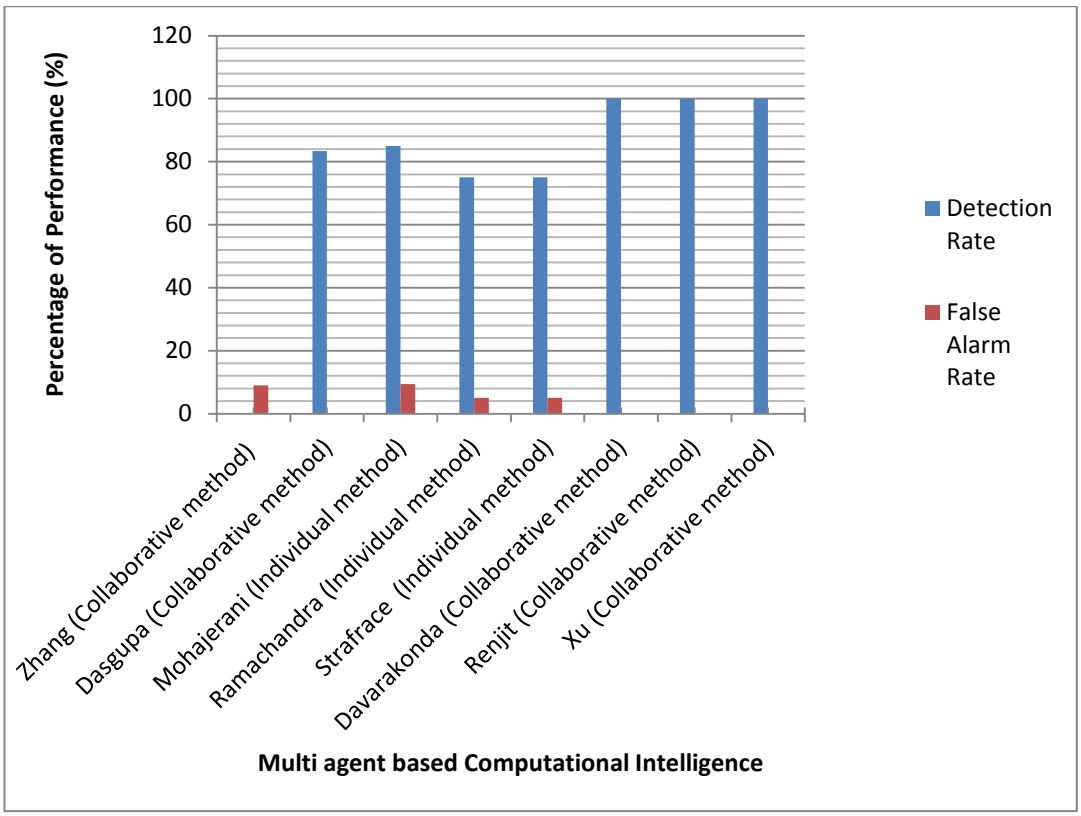

Figure 2.5: Comparison of Multi agent based CI methods in terms of detection rate and false alarm rate

Basically, at the moment, the majority of researchers are designing multi agent-based IIDPSs without integrating computational intelligence methods. In other words, CIs make use of MAS in terms of collaborative-based CI to optimize the functions of MAS.

2.3.4 Comparative discussion of detection rate evaluation

Because the environment, dataset, focus, scale, etc., in each experiment are totally different from scheme to scheme, detection accuracy and false alarm rate may not reflect the realistic performance. Therefore, these detection technique categories are vertically examined without the two factors. Table 2.9 shows the panoramic comparison of evaluations on the TAI, CI and MCI techniques.

Table 2.9: Panoramic comparison of evaluations on popular detection techniques.

Techniques		ACC	FAR	Remark
TAI	FCM (Tong et al., 2009) SOM (2011) GP (Phillips et al., 2010)	≈ 96% ≥ 80% ≥ 99.41%	≈1.5% ≈1.23% No Test	Simple and fast rule based techniques, performance is limited. Basically based on traditional artificial intelligence, good performance but complex.
CI	NF(Toosi et al., 2007) GF (Abadeh et al., 2007) MDP (X. Xu, 2010)	≈ 95% ≈ 99.08% ≈ 98%	≈1.9% ≈3.85% ≈1.2%	Basically based on Machine learning, balanced performance and complexity, high false alarm rate
MCI	RL (Vakili et al., 2011)	≥ 98%	≈0.98%	Good performance, based on computational intelligence, low false alarm rate

TAI and CI schemes own the strongest detection generality, as long as adequate attributes are in use. Their formidable capabilities of dealing with multiple-dimensional data fully support this to be realistic but what comes along with this capability is the high complexity. Fortunately, computational intelligence based detection in terms of using Multi agent may be implemented with the help of WSN's distributed architecture, which eventually cuts the complexity down as much as do those relatively advanced TAI techniques-based schemes. This kind of schemes is also characterized by the great flexibility, as it never depends on any prior-knowledge. The vertical evaluation on the three technique categories is illustrated in Table 2.10, where TAI stands for traditional AI, CI stands for computational intelligence, and MCI stands for multi agent based CI.

Table 2.10: Vertical evaluation of technique categories

Tech. category	Generality	Speed	Distributed	Prior knowledge
TAI	Low	Normal	Not	Assumption experience
CI	Normal t	High	Possible	Assumption
MCI	High	Normal	Necessary	Not

TAI: Traditional Artificial Intelligence; CI: computational intelligence; MCI: multi agent based computational intelligence

The computation complexity and memory use of TAI methods, such as rule-based detection schemes are lowest, indicating the fastest detection speed. However, they have to suffer from the weakest detection generality, since they are not equipped with the ability to dispose of multi-dimensional data. Inserting the new rules that cover more detection attributes into the rule set is the only way to push the detection generality up, which results in a linear increase of the complexity. The establishment of these schemes often demand some prior-knowledge regarding anomaly detection, either assumptions or experiences. The performance of traditional artificial intelligence techniques essentially stands in the middle. These schemes are enabled to be deployed in any WSN. Learning technique, such as reinforcement learning, is allowed to tackle multi-dimensional data, but the complexity would climb up dramatically. For this reason, multi agent system adapts to mitigate the problem of complexity by providing distributed sharing strategy.

Results of security evaluation metrics are compared through curves shown in Figure 2.6 and 2.7 respectively. The x-axis specifies the percentage of anomalous attack which refers to the ratio of the number of anomalous attack to the total number of measurements collected at the sensors. The y-axis specifies the security evaluation metric such as the accuracy of detection and false alarm rate.

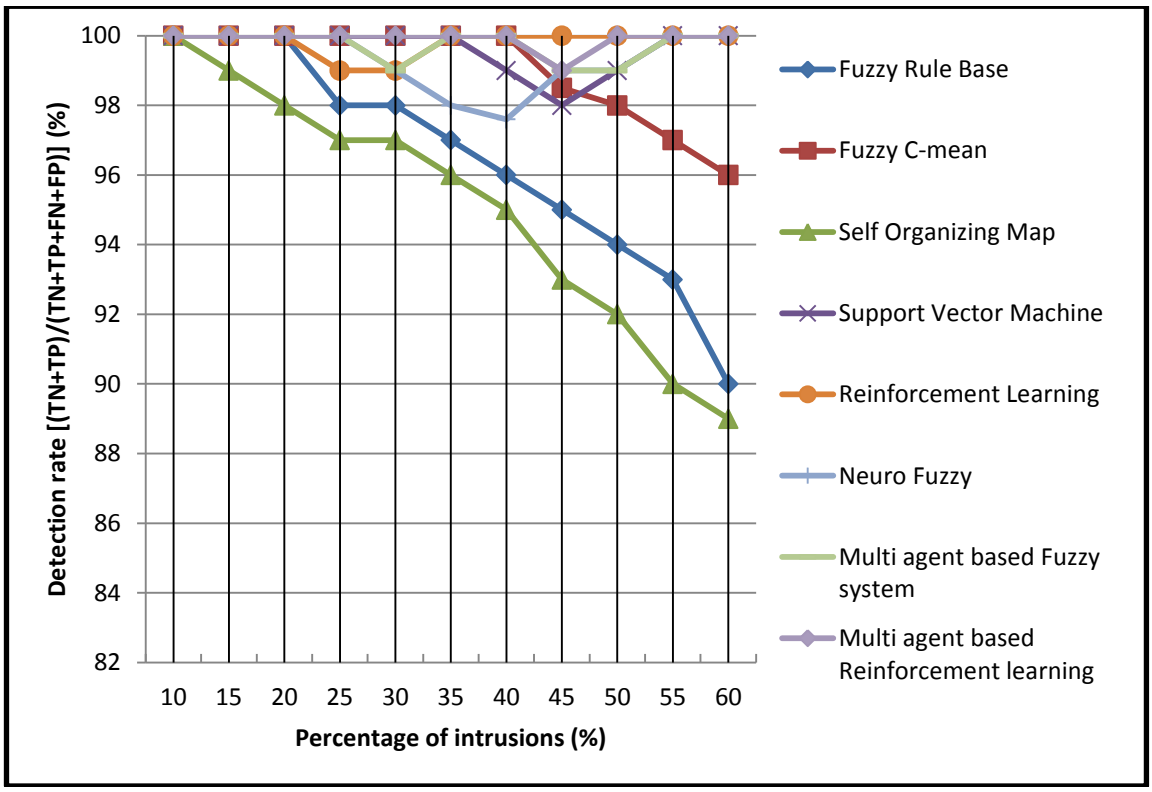

Figure 2.6: General comparison of detection rate

As seen in Figure 2.6, most of the studied algorithms successfully detect the anomalous attacks with a very close performance ratio up to 25% of increase in the anomalous data. Some of them, such as the fuzzy rule base and the self-organizing map rapidly deteriorate as the percentage of anomalous increase, while the superiority of the support vector machine intrusion detection technique, Fuzzy C-Mean clustering and Multi-agent based Reinforcement learning IDS, respectively, can be clearly observed for detection rate. They show better performance of detecting intrusions at 60% as shown in Figure 2.6.

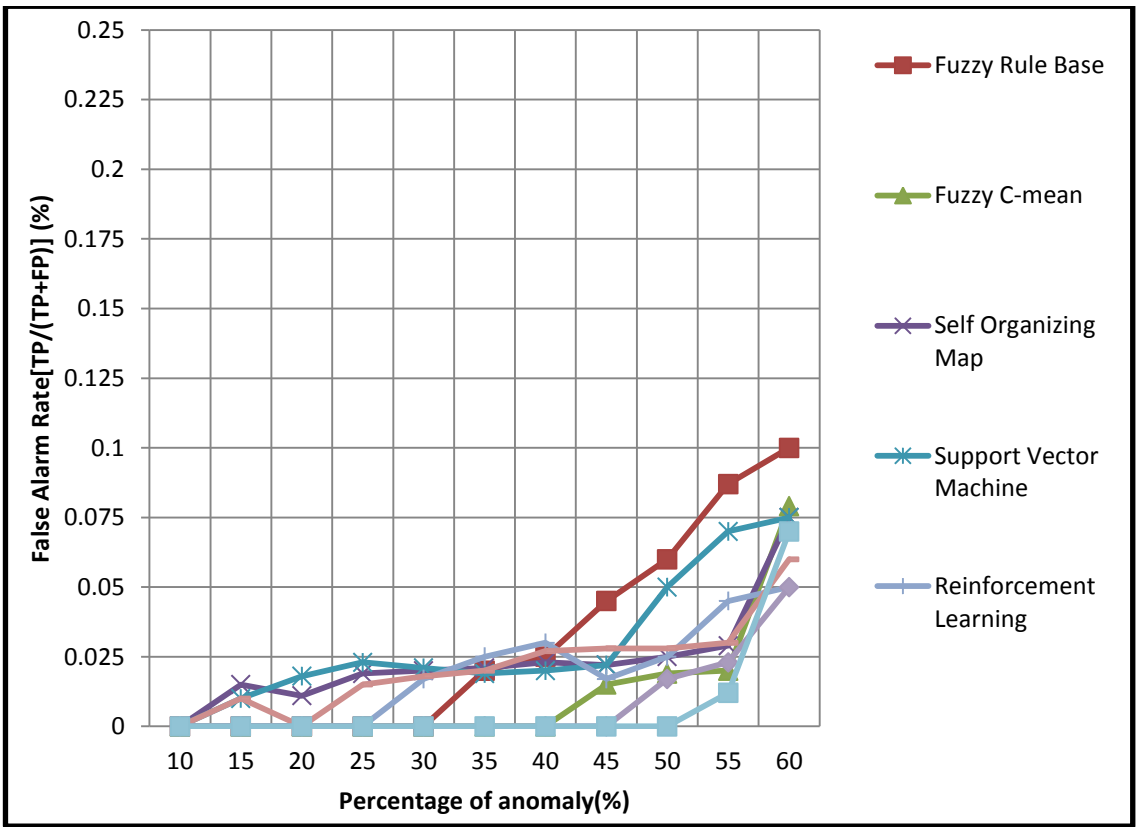

Figure 2.7: General comparison of false alarm rate

Regarding the false alarm rate, the comparison of the curves of the anomaly percentages is shown in Figure 2.7. Intrusion detection methods based on SVM classification, FCM clustering, and Multi agent refined reinforcement learning techniques have low false alarm rates as the percentage of anomaly increases.

2.4 Intelligent Intrusion Detection and Prevention System (IIDPS)

Intelligent techniques play a role in automating the intrusion detection process and to reduce human intervention. The process of intelligent detection applies advanced communication protocols based on artificial intelligence (AI) techniques such as fuzzy set, neural networks, and evolutionary computing, that operate as classifiers for anomaly detection to ensure detection accuracy along with stability (Idris et al., 2005). Denning (1987) used a rule-based expert system for Intrusion Detection Systems (IDSs) to improve detection performances. Although the rules may cover known patterns, they are unable to adapt in cases where attack patterns modify (e.g. attack polymorphs). In order

to provide high accuracy detection in anomaly detection, computational intelligence (CI) can serve in the construction of a model detection system by automatically iterating training and testing data. From our point of view, intelligent intrusion detection and prevention architecture for wireless networks contains four modules: (a) Matching stage, (b) Feature selection (c) Normalization, and (d) Decision (Figure 2.8).

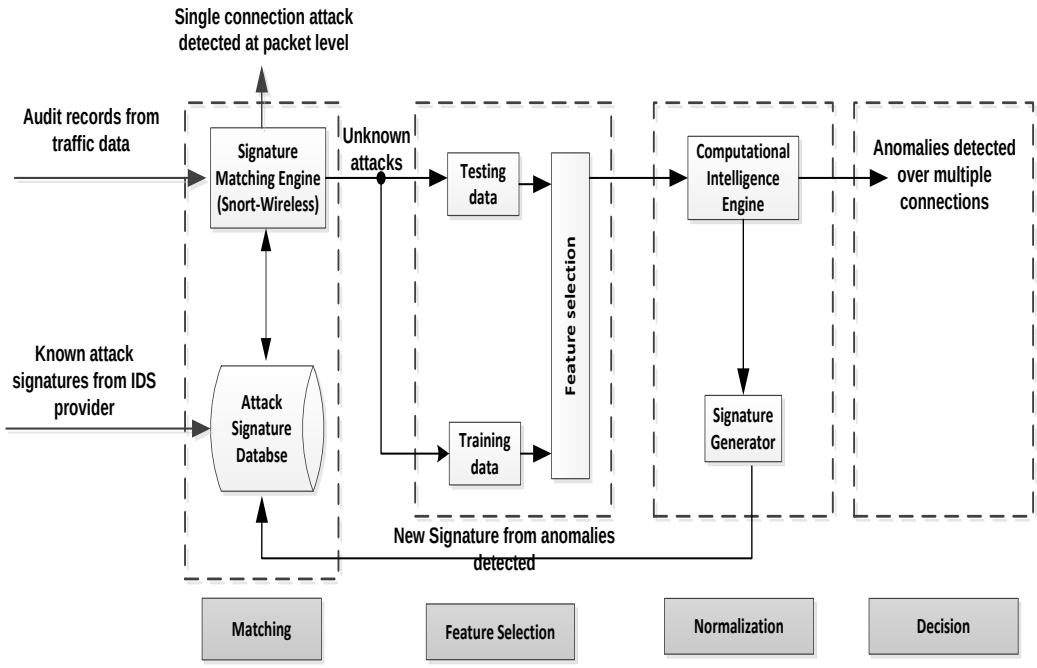

Figure 2.8: Intelligent Intrusion Detection and Prevention architecture for networks

In the general model of IIDPS, signature matching engine in terms of SNORT audit records from traffic data. The gain of matching the process includes two kinds of attacks: the single connection attack or known attacks detected at packet level and the unknown attacks. The process of detecting unknown attacks include training and testing algorithm and a corresponding model gets built through the feature selection module. The training and testing data are sampled first from the attack dataset. In addition, the feature selection method proposed is adopted to filter some unimportant and noise features to decrease the data dimension. The data are normalized through the normalization step, which are used to train the computational intelligence engine to make a model. The normalization module generates the signature of the matching module for inspection. In addition, the decision module is judged against the observed

traffic. If the deviation found exceeds (or is below in the case of abnormality models) a given alarm threshold, the detection stage is triggered.

IIDPSs are assessed on their capability to protect securely in large scale networks; nevertheless, utilization in a variety of networks and the complexity of the architecture (e.g. mobility, no central points, constrained bandwidth of wireless links, and limited resources) pose countless difficulties (Huang et al., 2013) .Some of the challenges remaining include questions as to how to reinforce the intrusion detections and response elements to deal with intrusion in parallel, in addition to the coordination and management of multiple nodes. Trust systems, like wireless network filtering facilities, focus on low-delay processing time, and high throughput performance.

Many of the preceding technical studies related to Intelligent Intrusion Detection and Prevention (IIDPS) methods were summarized and refined here (Abraham et al., 2007; Alan Bivens, 2002; 2011; Devarakonda et al., 2012; Ma et al., 2007; Renjit et al., 2011; Sevil Sen, 2011; Toosi et al., 2007) to bring a new perspective of IIDPS classification and the development for Cooperative-based IIDPS. Figure 2.9 lays out a tree plan classification of the anomaly-based IDPS detection techniques, and Table 2.11 outlines the advantages and disadvantages of detection in addition to the subtypes of detection-based IDPS. In the branch of IIDPS, the detection and prevention architecture uses Traditional Artificial Intelligence (TAI), which collects and analyses the information from single monitored classifiers (fuzzy sets, neural networks, genetic algorithms and artificial immune systems) in light of the availability of prior knowledge data. computational intelligence (CI) collects data from multiple monitored classifiers (neuro fuzzy, genetic fuzzy, Reinforcement Learning, Hidden Markov Model and Naïve Bayes) to detect entire, distributed and cooperative attacks, or “hybrids” of both, in terms of soft computing and machine learning approaches. Finally, Multi agent-based

CI (MCI) techniques are based on the establishment of a Multi agent model into soft computing and machine learning that allows the patterns analysed to be categorized.

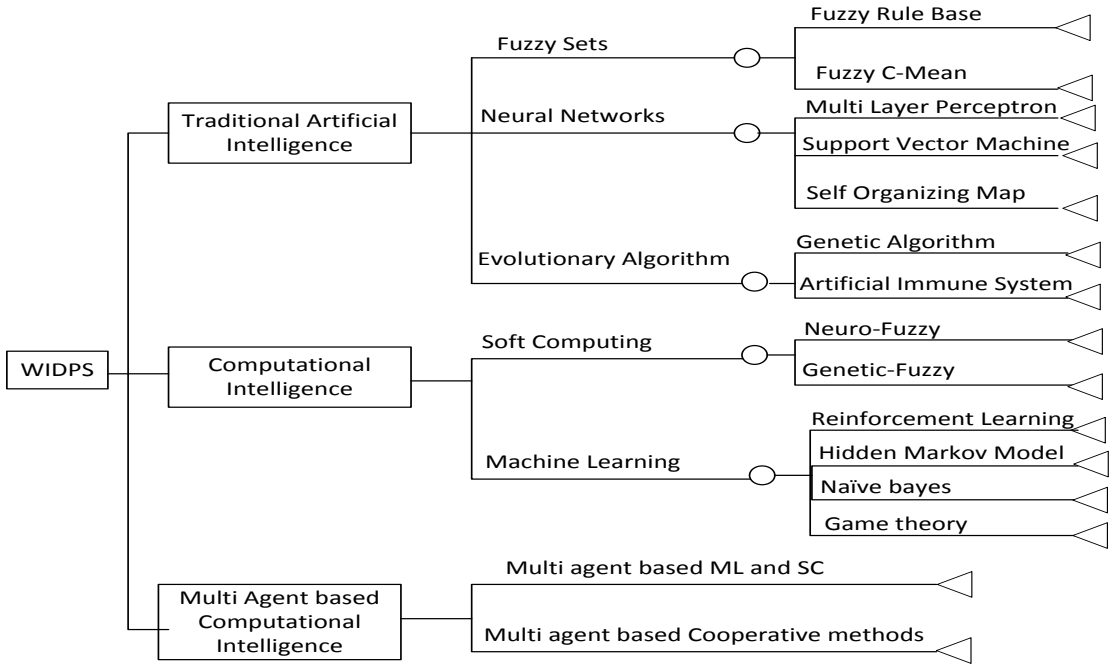

Figure 2.9: Tree plan classification of the anomaly-based IIDPS detection techniques

Table 2.11: Fundamentals of the anomaly-based IIDPS techniques

Main Types	**α. Pros ; β. Cons**	**Type of Detection**	**Subtypes of detection**
TAI: Availability of prior knowledge data	α. Robustness, flexibility and scalability β. Difficult setting for parameters and metric; time-consuming availability for high-quality data	Fuzzy Set: Approximation and uncertainty Artificial neural network: Human brain foundations Evolutionary computing: Biology inspired.	Fuzzy Rule Base, Fuzzy C-Mean Multi-Layer Perceptron, Self-Organizing Map, Support Vector Machine Genetic algorithm: Intrinsically parallel Artificial Immune system: Ability to converge very quickly
CI: Categorization of patterns	α. Flexibility and adaptability β. High resource and time consuming in training and testing stage	Soft computing: real-life situations Machine learning: A learner is to generalize from its experience	Neuro fuzzy, Genetic fuzzy: Universal approximate Hidden Markov Model: Autoregressive Bayes Naïve: Probabilistic relationships among variables Markov Decision Process: Stochastic Markov theory Reinforcement Learning: Dynamic approach applied to stochastic problem

			Game Theory: Modelling wide variety of situation
MCI: Cooperative pattern recognition	α. Robustness. Flexibility, adaptability and scalability.	MAS based computational intelligence: Categorization of patterns and Cooperative attempts	Clustering and outlier detection(Cooperative classification)

Table 2.11 provides the fundamentals for TAI, CI and MCI-based anomaly-IIDPS techniques, as well as the principal subtypes of detection. What is obvious from Table 2.11 is that the recent studies related to MCI-based IDPS concentrate on robustness, flexibility, adaptability and low resource consumption. For example, the highly accurate detection technique is one of the most favourable research areas regardless of other consequent challenges such as false alarm rate and response time. This chapter expands on the MCI based on collaborative techniques to help the Collaborative Intrusion Detection and Prevention (CIDPS) manager assimilate and synthesize false alarm rates into a well-managed set that applies to the whole networking environment under fully distributed collaborative management control.

2.5 Discussion

This thesis introduced three classes of IIDPS detection methodologies, approaches and technologies. Each technique has its advantages and limitations. The TAI-based IDPS is straightforward to implement and very effective in inspecting known attacks. Still, the approach hardly identifies unknown attacks, attacks concealed by evasion techniques and several variants of known attacks. A number of fuzzy rule-based approaches to detect unknown attacks were also proposed. Such techniques may unnecessarily result in issues with excessive computing time consumption and rapid updating of the knowledge base, hindering attack effectiveness.

A more accurate and simplified approach is still required to increase efficiency and effectiveness. Computational intelligence-based approaches such as Reinforcement Learning (RL) algorithm have the merit of possessing no prior knowledge of attacks. They do not work well in real-time applications due to the high computational complexity. A multi agent-based CI (MCI) not only mitigates high computational complexity such as time consumption and updating knowledge, but also enhances detection performance.

Table 2.12 tabulates the most recent multi-agent-based CI works applicable to IDPS. These are classified in terms of management structure, advantages and disadvantages. MCI-based IDPSs and the three concepts of management, namely FRLM, KM, and MA may be of assistance in designing an efficient system that satisfies Collaborative-IIDPS (Co-IIDPS) performance.

Table 2.12: Proposed Co-IIDPSs in terms of MCI methods classified according to our taxonomy

Reference	**Technology layout**	**Audit source location**	**Management structure**	**Advantage**	**Disadvantage**
Zhang et al. (2001)	ad-hoc networks	Public	Collaborative (MAS-CI)	Overcoming detection complexity by using distrusted agents	N/A
Mohajerani et al. (2003)	NIDS	Private	Individual (single)	N/A	The MLP neural network does not provide feedback.
Dagupta et al. (2005)	NIDS	Public	Collaborative (MAS-CI)	The advantage of having an individual agent for each functional module is to make future modifications easy	N/A
Agah et al. (2007)	CIDPS	Private	Collaborative (MAS-CI)	Using repeated decision policy significantly improve the chance of anomalous recognition	N/A
Mosqueira-Rey et al. (2007)	NIDS	Private	Individual (single)	N/A	N/A
Ramachandran et al. (2008)	ad-hoc	Public/Private	Individual (single)	N/A	N/A
Herrero et al. (2009)	NIDS	Private	Individual (single)	1. Scalability by adding new agents dynamically	The absence of a mechanism that

				anytime, 2.Failure tolerance by backup methods when working instances fail (proactive behaviour)	automatically responds, with the final decision made by the administrator.
Stafrace et al. (2010)	Wireless Ad-hoc	Public	Individual (single)	The collaborative work of the tactical squad of agents has led to concurrent detections of multiple threat	N/A
Renjit et al. (2011)	Wireless sensor	N/A	Collaborative (MAS-CI)	N/A	N/A
Leon et al. (2011)	Cognitive Radio	Private	Collaborative (MAS-CI)	Flexibility	N/A
Vakili et al. (2011)	CIDS	Private	Collaborative (MAS-CI)	Reliability as an impact factor hence learning mechanism	N/A
Fisch et al. (2012)	CIDS	Public	Individual	Broadcasting mechanism communication improved cooperative detection	N/A
Devarakonda et al. (2012)	HIDS	Public	Collaborative	N/A	NA
Patel et al. (2012)	CIDPS	Public	Collaborative	N/A	N/A

The features employed are very similar to each other. The most important varying features are management capabilities in the system structure. The collaborative management using the multi agent system-based computational intelligence portrays the ability to mitigate detection problems. In other words, the individual or single capabilities in terms of self-cooperative techniques (without using CI methods) consider all the features addressed in their systems. These inefficiencies are evidence of the lack of cooperative knowledge regarding suitable CI methods to identify intrusion prior to initiating any development.

All new solutions for developing multi agent-based CI methods consider the requirements (detection and false alarm rate) as being able to overcome Co-IIDPS complexities and meet the real operational goals of networks. As illustrated in Table 2.13 and as per our analysis, the proposed MCI-based Co-IIDPS mentioned in the

references meet two well-known requirements, and it is thus realistic to place them in actual network environments. The heterogeneous essence of network necessitates using MCI and individual techniques for Co-IIDPS to meet the stated requirements.

Table 2.13: The developed Co-IIDPSs (MCI) which met our proposed performance requirements

References requirements	True Positive	False Positive	False Negative
Zhang et al. (2001)	NMP	MR	MR
Mohajerani et al. (2003)	MR	MR	MR
Dasgupta et al. (2005)	MR	MR	MR
Agah et al. (2007)	MR	N/A	N/A
Mosqueria-Rey et al. (2007)	N/A	N/A	N/A
Ramachandran et al. (2008)	MR	MR	MR
Herrero et al. (2009)	N/A	N/A	N/A
Stafrace et al. (2010)	MR	MR	MR
Renjit et al. (2011)	MR	MR	MR
Leon et al. (2011)	P	NMP	NMP
Vakili et al. (2011)	P	NMP	NMP
Fisch et al. (2012)	P	NMP	NMP
Devarakonda et al. (2012)	MR	NMP	NMP
Xu et al. (2010)	MR	MR	MR
P = Partially NMP = not meet performance MR = meet requirement or performance N/A = Not applicable			

Incorporating a multi- agent system (MAS) to computational intelligence (MCI) in terms of Co-IIDPS allows monitoring intrusion activity. Fuzzy system (FS) with reinforcement learning (RL) in terms of fuzzy reinforcement learning manager (FRLM) has merged into Co-IIDPS, resulting in high true positive and low false alarm rates. The policy aspect of MAS-based FRLM applies a negotiation method to improve the detection accuracy. The developed Co-IIDPS around MAS-based FRLM satisfies the detection performance.

2.6 Chapter Summary

In this chapter, firstly, a comprehensive taxonomy along with state-of-the-art intrusion detection and prevention systems was presented. The scope was to capture researchers' attention into attempting to discover potential solutions to augment IDPS in order to minimize the impact of attacks on networks.

Secondly, the concept of intelligent intrusion detection and prevention system has been analysed in detail, showing the importance of this paradigm to enhance IDPS performance and reduce operational costs. In addition, within this broad concept, the main IIDPS were assessed and categorized into three trends: traditional artificial intelligence, computational intelligence and multi- agent-based computational intelligence.

Thirdly, this chapter shows the ability of multi agent based CI methods in terms of collaborative IIDPS. The conclusion is that further efforts are needed to find more effective solutions, especially those based on game theoretic-computational intelligence methods in terms of adaptive optimization techniques with cooperative approaches.

Chapter 3 : ADAPTIVE OPTIMIZATION TECHNIQUES

The first part of this chapter summarizes the principles of Fuzzy Logic, which is the theoretical foundation on which the techniques proposed in this thesis are based. The analysis focuses on FLCs which can effectively take a linguistic control technique that relies on expert knowledge and change it into an automatic control technique. The second part of the chapter is devoted to several techniques that are especially suitable for optimizing FLCs. Special attention is drawn to RL, which, in this thesis, is the method selected amongst the previously described techniques. The third part of the chapter encapsulates the principles of the Game Theory, which is the mathematical basis on which techniques proposed in this thesis are based. Special consideration is given to fuzzy reinforcement learning that adopts the Game Theory. In addition, a collaborative IIDPS based on fuzzy Q-learning is proposed.

3.1 Overview

In recent years, intrusion detection and prevention systems have become very important. To cope with security attacks on infrastructure over the last decade, security organizations have paid special attention to cost savings, with the concept of IDPS being of relevant interest (Pathan, 2014). From this perspective, self-optimization typically comprises network parameter tuning. Nonetheless, the set of network parameters that can be optimized in a network is extremely large, as there are countless IDPS algorithms running on it and their parameters need to be optimized. In addition, even if the optimization process is only done on a few relevant parameters, the connection among parameter settings and network performance is not clear-cut. For this reason, IDPS parameter optimization should be performed intelligently. As a result, the IDPS would be able to amend its parameters in terms of intelligence-based IDPS (IIDPS) in order to achieve optimum performance with no human work. However,

countless parameters can be changed remotely in the IIDPS system. From the operator's standpoint, adjusting IIDPS parameters that do not require time scheduling is the preferred alternative.

3.2 Fuzzy Logic

This section presents the theoretical basis of the computational intelligence methodology known as Fuzzy Logic. This discipline was initiated by Lotfi A. Zadeh (1965) , professor at the University of California, Berkeley.

Fuzzy Logic emerged as an important tool for system control and complex industrial processes, as well as for home and entertainment electronics, diagnostic systems and other expert systems. Currently, a multitude of applications based on Fuzzy Logic are applied in many different areas, for instance control systems, robotics, medicine, pattern recognition, computer vision, information and knowledge management systems, earthquake prediction, scheduling optimization, etc. As an alternative to Classical Logic, Fuzzy Logic introduces a degree of imprecision when items are evaluated (Precup et al., 2011). In real life, there is an abundance of knowledge that is ambiguous and imprecise, and human reasoning usually handles this kind of information. In this sense, Fuzzy Logic was designed specifically to imitate human behaviour. Additional benefits of Fuzzy Logic include simplicity and flexibility. In particular, this methodology can deal with problems with imprecise and incomplete data, and it can easily model non-linear functions of arbitrary complexity.

On the one hand, classical sets arise from the need for humans to classify objects and concepts. Such sets can be described as well-defined sets of elements or a membership function μ that can take a value of 0 or 1 from a universe of discourse for all elements that can belong (or not) to the concerned set. Formally, let X be the universe of discourse and x the elements contained in X. In addition, suppose A is a set

that contains some elements in the universe of discourse X. Then, the element x belongs or does not belong to set A, as characterized by the following function:

$$\mu_{A(x)}=\begin{cases}1 & if\ x \in A \\ 0 & if\ x \notin A\end{cases} \tag{3.1}$$

where $\mu_{A(x)}$ is the membership function corresponding to set A. Conversely, the necessity to work with fuzzy sets comes from the existence of concepts with no clear boundaries in their definition. Classical set theories categorize elements into crisp sets with well-defined boundaries between values. By contrast, fuzzy set theories classify elements into continuous sets based on an underlying theory that depends on the degree of membership. This means that membership functions are given a value ranging from 0 to 1 with undefined, gradual transitions between values. Formally, suppose B is a fuzzy set that contains elements in the universe of discourse X. Then such fuzzy set is characterized by the following membership mapping function:

$$\mu_{B(x):X\rightarrow[0,1]} \tag{3.2}$$

For all $x \in X$, $\mu_{B(x)}$ indicates how strongly Element X is connected to Fuzzy Set B. Although the membership function for a particular fuzzy set can be of any shape or type, an appropriate membership function is typically determined by experts in the field. In this sense, some membership functions, e.g. triangular, trapezoidal and Gaussian, are of special interest for designers.

As for crisp sets in Classical Logic, relations and operators can also be defined for fuzzy sets in Fuzzy Logic. In particular, these relations are the equality, containment, complement, intersection and union of fuzzy sets. Among these relations, the intersection of fuzzy sets plays a key role in designing rules for fuzzy controllers, as described in the next section. The intersection of sets A and B defines elements that occur in both sets. Operators that employ intersections are called t-norms. T-norms

results in sets that encompass all elements found in either Set A or Set B and also consider the degree of membership related to the t-norm. The most popular t-norms are defined as follows:

Min-operator.

$$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \forall x \in X. \quad (3.3)$$

Product operator.

$$\mu_{A\cap B}(x) = \mu_A(x)\mu_B(x), \forall x \in X. \quad (3.4)$$

As previously stated, Fuzzy Logic imitates human behaviour. One way of doing this is through using the notion of linguistic variables. A linguistic variable has a value of a word or sentence, allowing for computation with words rather than numbers. Such linguistic variable can be a word, linguistic label or an adjective. For example, let us consider the height of people in a country. In this case, the variable 'height' is a linguistic variable. A possible value for the numeric variable 'height' can be tall or short, meaning that a fuzzy set is associated with a linguistic term or value. In addition, certain adverbs can also be combined with adjectives to modify fuzzy values, e.g. very tall would refer to an individual who is noticeably taller than his peers. In other words, linguistic variables can serve to create numerical or logical statements from natural languages, facilitating handling human reasoning at the computational level. In this thesis, an attack data source can be defined as a 5-tuple ADS={Pt,Dp,Tr,Bs,Co}, or inputs of proposed algorithms according to the vulnerability scanning information, where Pt denotes the type of protocol (TCP=1, UDP=2); Dp is the destination port; Tr is the variance of time difference between two connections during a specific time window; Bs is the length of the packet from source to destination; and Co denotes the number of connections to the same host as the current connection in the past two seconds.

An important feature of Fuzzy Logic is that it provides a framework for handling rules (for control or decision making) that have earlier been expressed in an imprecise form. In this context, linguistic variables are embedded in an FLC's rules, facilitating the representation of human control expertise. More specifically, FLCs are composed of several IF-THEN rules that are easy to create. The succeeding section presents a short overview of FLCs, focusing on the components of such controllers and some types of fuzzy controllers for detecting features of distributed denial-of-service attacks.

3.2.1 Fuzzy Logic Controller Design

FLC design is one of the most important application areas of Fuzzy Logic (Engelbrecht, 2007). The main benefit of FLCs is that controlling a system (also called a plant) can be done using sentences rather than equations. This means that a control strategy can be described in terms of linguistic rules, in a more similar way to human language, instead of using for instance, differential equations. Since their start in 1975, many FLCs have been created for consumer products such as air conditioners, laundry appliances, audio visual equipment and industrial applications including hydro-electric generators, subways, and robotic controls. Over time, FLCs have proven they can provide better results than conventional control algorithms. FLCs are especially useful for complex processes that are beyond the scope of traditional quantitative methods, or when information is unreliable (Lee, 1990).

Designing an FLC includes defining the fuzzification and defuzzification processes, developing fuzzy control rules and generating a database. Figure 3.1 illustrates a generic FLC comprising four fundamental elements. The first element is the fuzzifier, which takes input data and changes it into linguistic variables that can also form labels for the fuzzy sets. Secondly, the knowledge base is a database and collection of linguistic statements based on expert knowledge, which is usually expressed in the

form of IF-THEN rules. Thirdly, the inference engine performs inference to compute a fuzzy output. Finally, the defuzzifier, which is the opposite of the fuzzifier, provides a non-fuzzy control action from an inferred fuzzy control action. The remaining paragraphs of this section describe each of these blocks in greater detail.

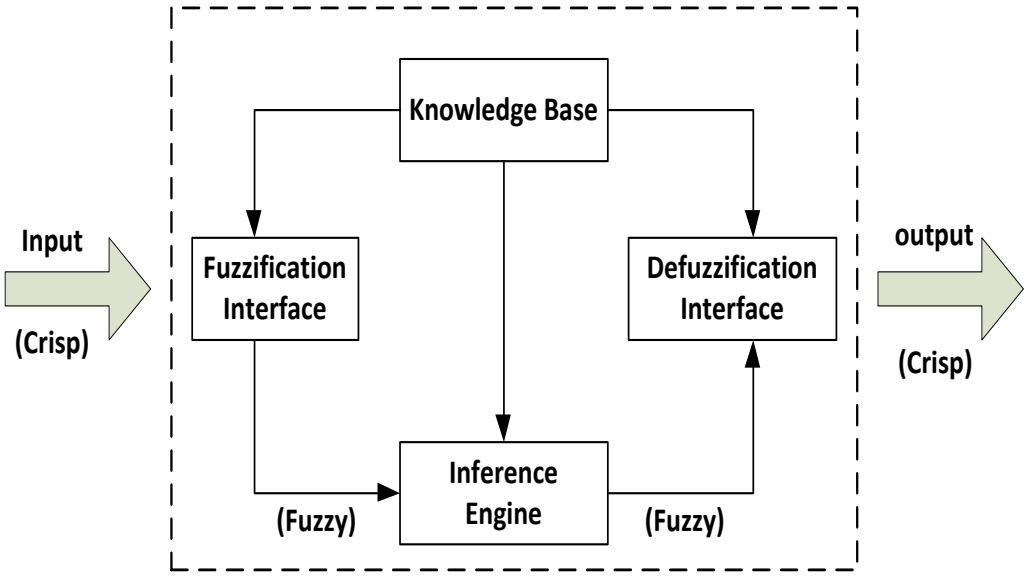

Figure 3.1: Block diagram of an FLC

Fuzzification process

The fuzzification interface begins by measuring the input variables' values. Next, a scale map is created that converts all these values into corresponding values from the universe of discourse. Afterwards, the non-fuzzy input values for the fuzzy representations are revealed.

In practice, the membership functions that correspond to each fuzzy set as determined in the input space are used to complete these tasks. More specifically, the fuzzification process is the assignment of membership values (one for each fuzzy value of the linguistic variable) to a numerical input value. For instance, let us consider the linguistic variable "time response," which can take fuzzy values of low, medium and high. Each input variable's sharp (crisp) value needs to first be fuzzified into linguistic

values prior to the fuzzy decision processes with the rule base. Formally, X denotes the universe of discourse for the three fuzzy sets. Hence, the fuzzification process receives the element $a \in X$, and produces the membership degrees $\mu_{low(a)}$, $\mu_{low(a)}$ $\mu_{medium(a)}$ and $\mu_{high(a)}$. The characteristic function of a fuzzy set is assigned values between 0 and 1, which represent the degree of membership of an element in a given set. Table 3.1 displays the linguistic terms and their fuzzy numbers used for evaluating the attack data source for time response, buffer size, and count. Figure 3.2 indicates the membership functions for time response.

Table 3.1: Fuzzy rating for the occurrence of attack traffic

Linguistic variables	**Fuzzy number**		
	Tr	**Bs**	**Co**
Low (L)	(-inf,-inf,0,40)	(-inf,0,2,3)	(-inf,0,1,1.5)
Medium (M)	(20,40,80,100)	(2,3,5,6)	(1,1.5,2,2.5)
High (H)	(80,120,inf,inf)	(5,6,8,inf)	(2,2.5,3,inf)

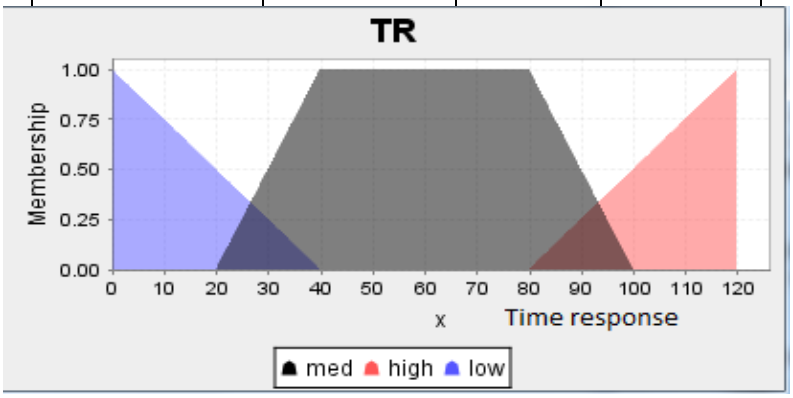

Figure 3.2: The membership functions of linguistic variables for attack data source Tr

Knowledge base

The FLC knowledge base comprises a database and a set of rules. The database permits fuzzy rules to be characterized and the fuzzy data to be manipulated. The set of rules provides the dynamic behaviour of the FLC through a set of linguistic rules derived from expert knowledge.

To begin with, the database is subjective because it is created from experience and judgments. The following aspects are related to database construction in an FLC:

- ***Discretization.*** Also referred to as quantization, its function is to convert a continuous universe into a discrete universe that contains a definite number of segments or quantization levels. In this case, membership values are assigned to generic elements found in the new discrete universe to identify a fuzzy set. In addition, there is a trade-off when selecting the number of quantization levels. On the one hand, it should be sufficiently large to provide appropriate granularity but on the other hand, it should be sufficiently small to save memory. In this sense, the corresponding mapping that transforms measured variables into values in the discretized universe can be linear, non-linear or both.

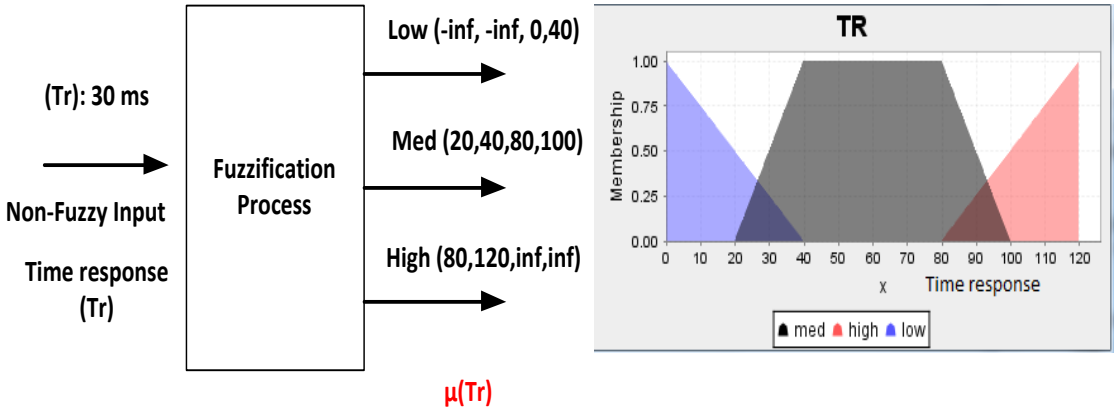

Figure 3.3: Example of fuzzification process

- ***Normalization.*** A universe of discourse is normalized when a discretization process is used to map a finite number of segments to their corresponding segments in the normalized universe. The mapping can be linear, non-linear or both.
- ***Partition of input and output spaces.*** A fuzzy partition determines how many fuzzy sets need to be defined and how granular the FLC control will be. This

depends on the characteristics of the system being controlled and the quality required for the control process.

- ***Completeness.*** The concept of completeness is related to the fact that the FLC generates an appropriate action for every state in the system. Typically, completeness regards design experience and engineering knowledge.
- ***Membership functions.*** A membership function determines the grade of the membership assigned to each fuzzy set. Decisions about these assignments are established using subjective criteria. For example, membership functions for input variables that are sensitive to noise are typically large enough to decrease that sensitivity. Membership functions are customarily expressed as bell, triangular or trapezoid-shaped functions.

  Secondly, the rule base is built using IF-THEN syntax to create control strategies as shown in Eq. (3.5):

IF (a set of conditions are satisfied) THEN (a set of consequences can be inferred), (3.5)

where the antecedent forms the first part of the conditional statement and the consequent if the second part. An antecedent is further defined as a condition of a domain. A consequent is a control action found in the system. The antecedents and consequents for a rule can contain more than one linguistic variable.

Defining how an FLC will be characterized depends on selecting the state variable form the antecedent and control variables found in the consequent. An instance of a rule is “if pressure is very high, then open the valve.” One of the benefits of using these rules is that they characterize human behaviour and can be used to analyse decisions since they supply a framework. In this sense, according to several researchers domain knowledge can easily be communicated using fuzzy control rules. To formulate these fuzzy rules, operators and experts in this field were queried using a carefully

organized questionnaire. This explains the fact that FLCs are implemented using fuzzy IF-THEN rules.

Inference engine

Once the input variable values have been converted to fuzzy values through the fuzzification process, the inference engine identifies which rules are triggered and calculates the fuzzy values of the output variables. In other words, this process connects the rule base to the fuzzified inputs to develop the fuzzified output for a rule. To do this, each output set must be assigned a degree of membership that is part of the consequents in the fuzzy rules. This is calculated using the degree of membership found in the input sets in addition to the affiliation between input sets. These connections are established using a logic operator, which takes sets from the antecedent and combines them. Then, the output fuzzy sets from the consequent are added to create one general membership function that will act as the output for the rule.

To explain the inference process, assume that A and B are two input fuzzy sets in the universe of discourse. The X2 universe of discourse includes a fuzzy set with X1 and C. Let us also consider that the following rule is defined:

$$IF\ (A\ is\ a\ and\ B\ is\ b)\ THEN\ (C\ is\ c) \tag{3.6}$$

The $\mu_{A(a)}$ and $\mu_{B(b)}$ values are available to the inference engine because a fuzzification process was used in their development. Thus, the inference process starts by taking a rule base and calculating the degree of truth for each rule. The degree of truth specifies the triggering strength of a particular rule. It is calculated by combining the antecedent sets using specific operators, among which the min-operator and product operator for the intersection relation as previously stated. In this example, assuming the min-operator, the degree of truth _*k* for rule *k* is calculated as:

$$\alpha_k = min\{\mu_A(a), \mu_B(b)\}. \quad (3.7)$$

The following step in the inference process is to determine a single fuzzy value for each output $c_i \in C$ that has been activated. In general, the final fuzzy value corresponding to the output, c_i , denoted as β_i, is computed using the max-operator as follows:

$$\beta_i = max_{\forall k}\{\alpha_{k_i}\} \quad (3.8)$$

where α_{k_i} is the degree of truth of rule *k*, which activated output c_i.

The final result of the inference engine is a set of fuzzified output values. In this case, the rules that are not activated have a degree of truth equal to zero. In addition, rules can include a weighting factor in the range [0, 1] to represent the degree of confidence in that rule. Such factors derived from expert knowledge are applied when the fuzzy rules are aggregated to produce a non-fuzzy value in the defuzzification process.

Defuzzification process

This method establishes a relationship between fuzzy control action spaces, the outputs from the universe of discourse, and crisp, non-fuzzy control action spaces. In the consequent, a set's degree of membership is represented by a rule's degree of truth. Given the degree of truth from a set of activated fuzzy rules, the defuzzification process creates non-function scalar values from the output of fuzzy rules. To calculate such scalar values, two different approaches can be used. Mamdani et al. (1975) developed a fuzzy rule that was the foundation for the first approach, where the rules lead to a consequent that is another fuzzy variable [see (3.6) as an example]. The second approach is known as the Takagi-Sugeno approach and it uses rules with consequents that are polynomial functions of the inputs (Takagi et al., 1985).

The Mamdani approach:

In this method, there are several ways to find a scalar value that represents what actions will be taken.

• **Maximum-Minimum Approach:** In this approach, rules with the highest degree of truth are selected. Then the membership functions of the activated consequents are determined. Finally, the centroid for the area covered by the membership function is found. The FLC's output is the centroid's horizontal coordinate.

• **Averaging Approach:** The averaging approach uses the average of the degrees of truth for all activated rules. After the average is calculated, the membership functions are limited to this average. Next, the horizontal coordinate of the centroid for the area is determined and used as the FLC output.

• **Root-sum-square method.** The membership functions are rated such that the apex for each function is that same as the maximum and the peak of each function is equal to the maximum degree of truth value associated with that particular function. As in the averaging approach, the horizontal coordinate of the centroid for the area is calculated to form the FLC output.

• **Clipped Centre of Gravity Method:** In this method, the membership functions are shortened, or "clipped" so they are equal to the degree of truth for the corresponding rule. The next step is to find the horizontal coordinate of the centroid for the area, which will also be used as FLC output.

Calculating the centroid for a trapezoidal area depends of whether the domain of the membership functions is continuous or discrete. A finite amount of values, n_x, are found in a discrete domain and the following equation is used to calculate the defuzzification process results:

$$output = \frac{\sum_{i=1}^{n_x} x_i \mu_C(x_i)}{\sum_{i=1}^{n_x} \mu_C(x_i)} \quad (3.9)$$

where x_i is each possible value. In the case of a continuous domain, the output is given by the following expression:

$$output = \frac{\int_{x \in X} x\mu(x)\, dx}{\int_{x \in X} \mu(x)\, dx} \quad (3.10)$$

where X is the universe of discourse.

Takagi-Sugeno approach:

A typical rule for this approach adheres to the following generic expression (Takagi et al., 1985):

$$\text{IF } (X_1 \text{ is } A_1 \text{ and} \ldots \text{and } X_n \text{ is } A_n) \text{ THEN } (Y = P_0 + P_1 X_1 + \ldots + P_n X_n). \quad (3.11)$$

where X_1,..., X_n represent the fuzzy input variables and A_i indicates one of the fuzzy sets for the linguistic variable X_i; Y denotes the output variable; and P_0,..., P_n are the parameters. Thus, the main difference between the Takagi-Sugeno and Mamdani approaches is that in one of them, the consequent of the rule is a mathematical function instead of a fuzzy consequent. Furthermore, the Takagi-Sugeno has been extended to non-linear functions. When a set is composed of a set of activated rules and associated degrees of truth, calculating the resulting crisp value as a weighted average of the rule outputs can be done with the following equation:

$$output = \frac{\sum_{i=1}^{N} \alpha_i . f(X_1, \ldots, X_n)}{\sum_{i=1}^{N} \alpha_i} \quad (3.12)$$

where N represents the number of rules and $f(X_1, \ldots, X_n)$ signifies a few of the mathematical input functions.

The main benefits of the Takagi-Sugeno approach are that a more dynamic control is provided, FLCs are computationally more efficient and best suited for mathematical analysis, and it works well with optimization and adaptive techniques. For these reasons, the FLCs proposed in this thesis are based on the Takagi-Sugeno approach.

To conclude this section, an illustrative example of FLC operation is provided. The FLC is based on the Takagi-Sugeno approach explained previously. Suppose that the controller is described by the following two rules:

$$\text{IF } (x \text{ is } A_1 \text{ and } y \text{ is } B_1) \text{ THEN } (z \text{ is } f_1(x,y) = K_1) \quad (3.13)$$

$$\text{IF } (x \text{ is } A_2 \text{ and } y \text{ is } B_2) \text{ THEN } (z \text{ is } f_2(x,y) = K_2) \quad (3.14)$$

from which the following elements can be identified:

- Variables x and y represent the universe of discourse X and Y, respectively.
- Two fuzzy sets, A1 and A2, are defined for variable x.
- Two fuzzy sets, B1 and B2, are defined for variable y.
- There is one output variable, z.
- Two constant functions, f_1 andf_2 , are defined for variable z.

The membership functions defined for each fuzzy set of input variables are shown in Figure 3.4.

- Then, the basic FLC operation is as follows:
  - Step 1. The fuzzification process calculates the membership value for each fuzzy set by applying the associated membership function as shown in Figure 3.5 (a).
  - Step 2. The inference engine computes the degree of truth for each fuzzy rule through the combination of fuzzified inputs using the min-operator, as shown in Figure 3.5 (b). The expressions used to calculate the degree of truth are:

$$\alpha_1 = min\{\mu_{A_1}(x_0), \mu_{B_1}(y_0)\} \quad (3.15)$$

&

$$\alpha_2 = min\{\mu_{A_2}(x_0), \mu_{B_2}(y_0)\} \quad (3.16)$$

- Step 3. Finally, the defuzzification process calculates the non-fuzzy output as a weighted average of the rule constant outputs. The equation to produce the output value is:

$$output = \frac{\alpha_1.k_1+\alpha_2.k_2}{\alpha_1+\alpha_2} \quad (3.17)$$

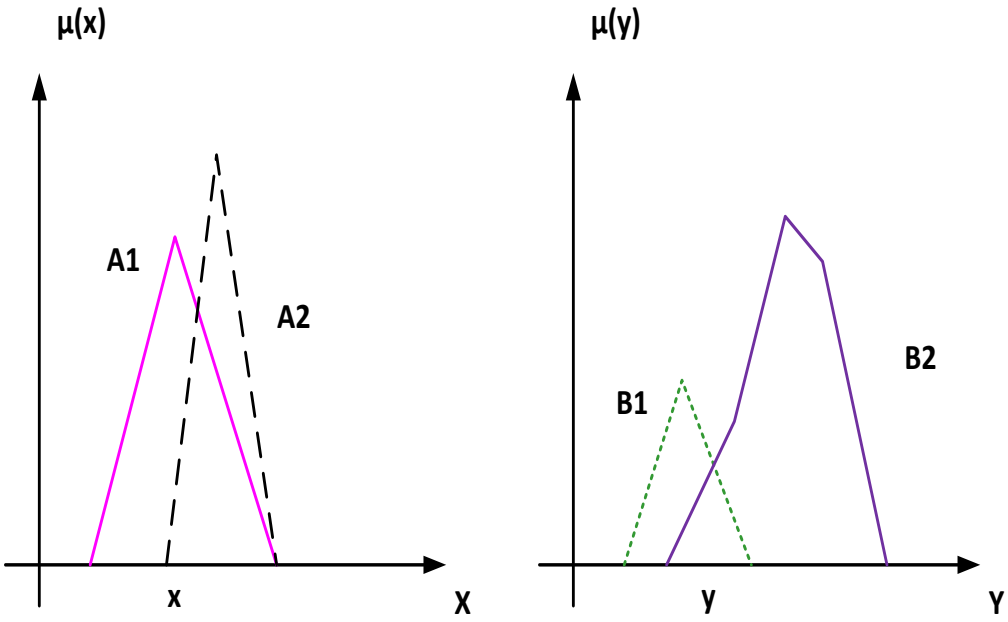

Figure 3.4: Membership functions of the input fuzzy sets as example

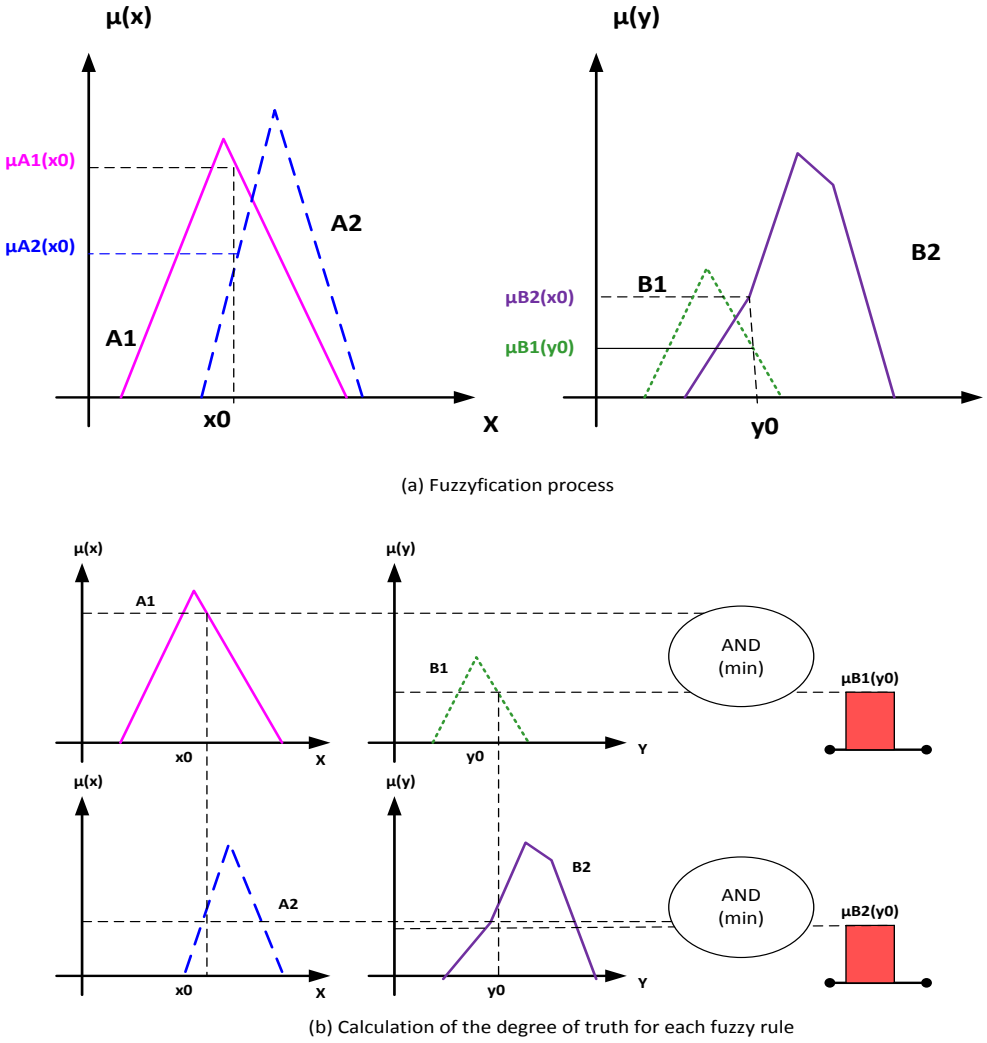

Figure 3.5: Basic FLC operation example

Optimization of the self-tuning process

An important area in algorithmic development to cope with complex problems is the design of so-called intelligent algorithms. In this context, the development of models based on biological and natural intelligence has played a key role in the last years. Artificial neural networks, reinforcement learning, evolutionary computation, and swarm intelligence are all examples of such algorithms. More recently, several of these approaches have been combined with each other or with traditional methods to solve challenging and complex issues. Moreover, these algorithms are part of the field of machine learning employed in several different areas of research, including many social

sciences and computer science. In principle, any of these approaches should be capable of yielding results in a relatively short time.

In this thesis, the proposed auto-tuning algorithms are based on FLC design. Since detecting DDoS attacks in networks can be considered complex due to the presence of a large multitude of diverse and interacting elements, defining the controller usually becomes a challenging task for IIDPS. In addition, as DDoS attacks in networks involve unpredictable and highly variable context factors, which can also vary on different time scales, the FLC needs to be reconfigured throughout adjusting the detector engine parameters in order to identify those traffic variations.

Thus, to cope with the dynamic variations, the lack of knowledge or, simply, to refine the behaviour of the controller, different strategies have been analysed in this thesis, namely two reinforcement learning algorithms (i.e., Sarsa and Q-learning). The objective of these mathematical techniques is to optimize the behaviour of the FLC through a learning process. In this section, after providing a general overview of RL techniques, a more detailed overview is devoted to the Q-learning algorithm and how FLC optimizes with Q-learning, which is of particular interest in scenarios such as wireless networks in which learning from interaction becomes essential for detecting DDoS attacks. In this section, an optimized IIDPS is proposed, which utilizes the fuzzy Q-learning algorithm with weighted strategy sharing in terms of multi-agent system-based IIDPS. The analysis particularly stresses on the cooperative game theory-based fuzzy Q-Learning algorithm, which is a promising approach in the context of cooperative IIDPS in this thesis.

3.3 Reinforcement Learning

In a particular environment, an agent can be encouraged to engage in a specific action that will lead to maximizing a cumulative reward. This type of machine learning is known as RL. Its two defining characteristics are a trial and error search and actions with consequences that can affect immediate and future rewards (Sutton et al., 1998).

RL differs from other learning approaches, such as supervised learning that is typically used in Neural Networks. In the latter case, learning is done by using previously collected examples or sets of training data, which are not appropriate for interactive learning. In addition, it is difficult to find a training data set that adequately represents all of the situations in which an agent would be required to act. Thus, in those instances, an agent that can learn from its own experience remains the only answer.

Beyond the agent and environment, the following elements can be identified in RL:

- ***Policy*:** the policy defines how the agent must act at a given time. In other words, it connects perceived states from the environment and actions to be taken as a result of those states.
- ***Reward Function:*** the reward function describes the goal of an RL problem. More specifically, it is a map between each perceived state and a scalar or reward that sets out the value of being in that state. However, the objective of the agent is to maximize the total reward rather than the immediate reward. The reward function signifies immediate value.
- ***Value Function:*** the value function specifies what is good over the long term. In particular, the value function is a map between each perceived state and the rewards that an agent can expect to accumulate over time beginning with a particular state.

- ***Model of the Environment (optional):*** the model of the environment illustrates how the environment behaves. RL is capable of learning by trial-and-error while at the same time learning a model of the environment.

To illustrate these concepts, a basic scheme of an RL problem is shown in Figure 3.6 where a general environment responds at time *t + 1* to an action taken at time *t*.

A key concept in RL is the trade-off between exploration and exploitation. When an agent is required to act, it will select an action that has yielded rewards in the past. However, in the absence of former results, the only way to discover what actions will be profitable is to try actions that have not been previously selected. In other words, the trade-off between exploration and exploitation rests on an agent's ability to take advantage of current knowledge but remain open to other, untried actions. In this sense, the agent's primary goal is to maximize the rewards achieved over the long term, that is, the sum of the rewards obtained from all situations or states that will be visited in the future:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots . = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (3.20)$$

where r represents the consequence of an action that leads to a numerical reward for each time step and *Y* denotes the discount rate given to indicate how important a future reward will be.

The Markov property

As indicated previously, the function of a state influences how an agent will make decisions. In this context, important environmental properties and state signals, otherwise known as the Markov Property, can be found. The state signal includes all the information available to the agent. However, the agent does not expect to receive any information that would facilitate decision making or even all the information regarding the environment.

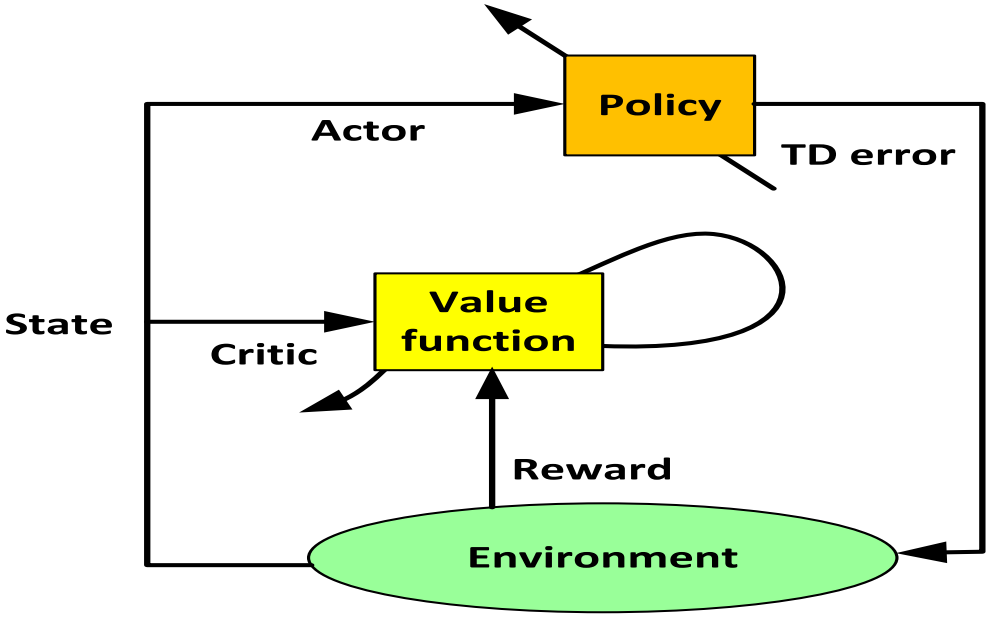

Figure 3.6: The basic elements of an RL problem

An appropriate state signal is one that summarizes past information compactly, but also maintains the relevant information parts. The Markov property is fulfilled when a state signal retains all relevant information. In this situation, at time step *t+1*, the response of the environment is only dependent on time *t*. As such, the environmental dynamics can be defined as:

$$P_r\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}, \quad (3.21)$$

where $P_r\{.\}$ denotes the probability of its argument; s is the state of the environment; s' is any state in the system; r is the received reward; and a denotes the action taken by the agent. When the environment contains the Markov property, it is possible to predict the next state and rewards based on current states and actions.

A Markov decision process (MDP) is an RL task with the Markov property. A finite MDP has finite states and actions and is further defined by a set of actions and states and the dynamics of the environment. The latter is specified by transition prospects and how valuable the next reward is expected to be. The transition probability

for each probable next state (s') for any current state (s) and any action (a) can be calculated using the equation below:

$$P_{ss'}^{a} = P_r\{s_{t+1} = s' | \, s_t = s, a_t = a\}. \quad (3.33)$$

Likewise, the expected value of the next rewards for any current action a, state s, and next state s', is calculated as follows:

$$R_{ss'}^{a} = E\{r_{t+1} | \, s_t = s, a_t = a, s_{t+1} = s'\} \quad (3.34)$$

where $E\{\cdot\}$ is the expected value of its argument. The most important factors of a dynamic finite MDP are its transition probabilities and expected value of the next reward.

Optimal value functions

Most RL algorithms search for value functions that assess the benefits of a given state available to an agent. As previously stated, the expected accumulated reward measures the value of the state s. In RL, a state-value function, called $V(s)$, is used to identify the benefits of obtaining state s. The value is subject to what states the agent has visited, which in turn depends on the what policy has been followed. A policy function π is a map that shows the connection between states and actions used to govern how the agents will behave. In contrast, $\pi(s, a)$ indicates the likelihood of engaging in action a from state s. In this case, the value of state s following policy π is defined as:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}$$

$$= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s\}, \quad (3.24)$$

where $E\pi\{\cdot\}$ means the expected value under policy π.

Similarly, in RL, the action-value function $Q(s,a)$ qualifies the value of taking an action a when starting from state s. If the agent follows policy π, then it is formally expressed as:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t |\, s_t = s, a_t = a\}$$
$$= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}. \quad (3.25)$$

Experience can be used to estimate functions V^{π} and Q^{π}. A fundamental property of these functions is that they meet the requirements of certain recursive relationships. In other words, the following condition holds between the value of s and possible successor states for any policy π or any state s:

$$V^{\pi}(s) = E_{\pi}\{R_t |\, s_t = s\}$$

$$= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

$$= \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\}\right]$$

$$= \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^{\pi}(s')\right], \quad (3.26)$$

Equation 3.26 is also called the Bellman equation for V^{π}. Moreover, the solution to this equation is the value function V^{π}.

Finding a good policy that will result in long-term rewards is the same as solving an RL problem. An optimal policy always has an expected value greater than (or equal to) other policies for all states. Likewise, the best policies have equal state and action value functions called $V^*$and Q^* respectively. $V^*$is expressed as:

$$V^*(s) = max_{\pi} V^{\pi}(s), \quad (3.27)$$

for all $s \in S$ when S represents the set of states. Similarly, Q^* is described as:

$$Q^*(s,a) = max_{\pi}(s,a), \quad (3.28)$$

for every $s \in S$ and $a \in A(s)$, where $A(s)$ indicates the set of possible actions in state s. Function Q^* delivers the return expected as a result of an action in state s before it follows an optimal policy. This process can be stated in terms of V^* as demonstrated below:

$$Q^*(s,a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}. \quad (3.29)$$

The Bellman equation for V^* can be rewritten without making reference to any specific policy. If this happens, it is known as a Bellman optimality equation. Bellman optimality equations state that when the best action is taken from a state, its expected return is the same as the value of the state under an optimal policy, as shown below:

$$V^*(s) = max_{a \epsilon A(s)} Q^{\pi^*}(s,a)$$

$$= max_a E_{\pi^*}\{R_t | s_t = s, a_t = a\}$$

$$= max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

$$= max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s')\right]. \quad (3.30)$$

The Bellman optimality equation for Q^* is:

$$Q^*(s,a) = E\{r_{t+1} + \gamma max_{a'} \gamma\, Q^*(s_{t+1}, a') | s_t = s, a_t = a\}$$

$$= \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma max_{a'} Q^*(s', a')\right]. \quad (3.31)$$

The Bellman optimality equation is a series of equations where each state is represented by its own equation. In other words, N equations will represent N states with N variables. In terms of finite MDPs, this means that the solution for the Bellman optimality equation is independent of policy. Furthermore, any techniques used to find

solutions for systems in non-linear equations can be used in cases where the dynamics of an environment ($P_{ss'}^{a}$ a or $R_{ss'}^{a}$) are known.

Discovering an optimal policy becomes fast and easy once the system of equations is solved. When the value of $V^{*}$is known, the best actions in the next step become optimal actions and any greedy policy becomes an optimal policy.

V^{*} is significant because it considers the rewards generated from all future behaviours. When it is used to assess short-term consequences that result from an action, it establishes a greedy policy that is optimal in the long term.

Alternatively, if the value of $Q^{*}$is known, the agent is not required to find the actions for the next step. Instead, it only looks for actions that maximizes Q^{*} (s, a). In these cases, choosing the best options becomes even easier. In other words, the optimal action-value function does not require information about possible successor states and values, or the dynamics of an environment to determine optional actions.

Solving the Bellman optimality equation creates a method of discovering optimal polices and solving RL problems. Unfortunately, these solutions are not useful without further adjustments. In practice, three assumptions must be made: (a) accurate knowledge regarding the dynamics of the environment, (b) sufficient computational resources to find solutions, and (c) the Markov property.

To solve problems in an approximate way, many different decision-making methods can be applied, for example heuristic search methods and dynamic programming. In this context, many RL methods are clearly viewed as approximate means to find solutions for Bellman optimality equations. In such instances, real, experienced transitions are employed rather than knowledge about expected transitions.

The techniques most commonly employed for solving RL problems are Monte Carlo, temporal different methods and dynamic programming. Each class of methods has advantages and disadvantages. Dynamic programming methods, which attempt to solve Bellman equations, are successful because they are mathematically sound, but they require a complete and accurate model of the environment. Monte Carlo methods attempt to estimate value functions and discover optimal policies. They are conceptually simple and a model is not required, but they do not function for calculations that require step-by-step processes because they use averaging sample returns and work best for episodic tasks. To overcome this limitation, experiences are divided into episodes. When an episode is completed, then the policies and value estimates are modified. Finally, temporal-difference methods require complex analysis but are fully incremental and a model is not required. These three different method types also vary in terms of efficiency and speed of convergence, and they can be combined in order to obtain the benefits of each one.

Q-Learning algorithm

Mechanisms for determining optimal policies follow generalized policy iterations based on alternating policy improvements and evaluations. Policy evaluation is used to make value functions resemble current policies. Policy improvements utilize new value functions to enhance policies in terms of expected value. This concept is illustrated in Figure. 3.7. The result of such an iterative process is that both policy and value functions approach optimality.

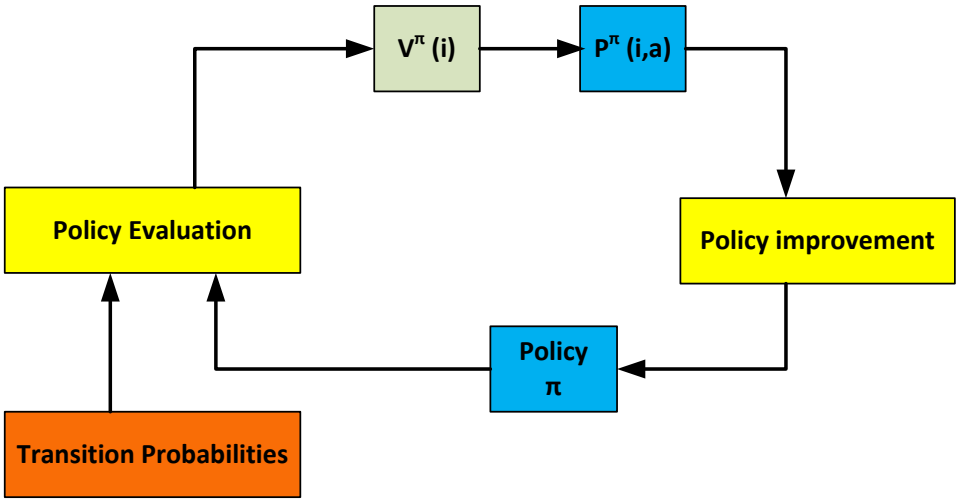

Figure 3.7: Basic scheme of generalized policy iteration

An action-value function is used instead and is calculated by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \quad (3.33)$$

Policy improvement is achieved by selecting actions whose current action-value is the greatest in that state, meaning to make the policy greedy by:

$$a(s) = argmax_k Q(s, k). \quad (3.34)$$

If all state-action pairs can be visited an infinite number of times, the limit of the policy becomes greedy and the process converges to the optimal value function and policy.

Watkins et al. (1992) defined Q-learning as a temporal-difference algorithm where the learned Q(s,a) is a direct approximation of the optimal Q*(s,a) regardless of the policy followed by the agent. In order to calculate the updated action-value function, the following equation is used:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[r_{t+1} + \gamma max_k Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (3.35)$$

where Q is the optimal action function Q* without depending on the policy followed.

3.4 Q-Learning Adaptation to FLCs

Applying RL techniques to an FLC rule base will lead to optimization. A Q-Learning fuzzy version was developed by Glorennec (1994) to optimize the consequent of the fuzzy rules found in an FLC. Q-Learning permits continuous state and action spaces when the action-value function is discretized. The resulting discrete q values are stored in a look-up table as a finite set of state-action yards. Another benefit of using this method is that the fuzzy rules easily accept prior knowledge and the learning process becomes faster.

The agent is forced to select an action from J for rule i when the action space is discretized. In some situations, the FLC has N fuzzy rules and $a[i,j]$ forms the j^{th} possible action for rule i and $q[i,j]$. Any associated q-values are kept in the look-up table. Ultimately, representations of continuous $Q(s,a)$ are considered to be the same so that the q-value of each rule consequence can be determined before being used in the continuous input vector. The steps involved in the fuzzy Q-Learning algorithm are as follows:

1. In the look-up table, initialize the q*values. If no prior knowledge is available, use the following equation:

$$q[i,j] = 0, 1 \leq i \leq N \; and \; 1 \leq j \leq J, \tag{3.36}$$

where $q[i,j]$ represents the q-value, N signifies the number of rules and the number of actions for each rule is represented by J.

2. Select an action for each activated rule that has a nonzero degree of truth. Actions can be selected using one of the following $\epsilon - greedy$ policies:

$$a_i = argmax_k \; q[i,k] \;\; \text{with probability 1-}\epsilon, \tag{3.37}$$

Or,

$a_i = random\{a_k, k = 1,2, \dots, J\}$ with probability ϵ, (3,38)

In Equations 3.37 and 3.38, a_i represents the consequent of rule i and ϵ signifies the parameter that establishes the trade-off between exploration and exploitation in the algorithm. In other words, $\epsilon = 0$ indicates that the best action was selected and there was no exploration.

3. Determine the global action suggested by the FLC using the equation recommended below:

   $a(t) = \sum_{i=1}^{N} \alpha_i\big(s(t)\big).\ a_i(t),$ (3.39)

where $a(t)$ denotes the inferred action at time step t, $\alpha_i(s(t))$ represents the degree of truth for rule i and $a_i(t)$ indicates the selected action for rule i. The degree of truth is the distance between rule i and input state $s(t)$ and is computed as:

$\alpha_i\big(s(t)\big) = \prod_{j=1}^{L} \mu_{ij}\left(s_j(t)\right),$ (3.40)

where the total amount of FLC inputs is represented by L and the membership function is represented by $\mu_{ij}\left(s_j(t)\right)$ for the j^{th} FLC input and rule i.

4. The Q-function from the current q*-value and degree of truth for the rules is calculated as:

$\mathrm{Q(s(t), a(t)) = \sum_{i=1}^{N} \alpha_i\big(s(t)\big).\, q[i, \alpha_i]},$ (3.41)

where the value of the Q-function is expressed as $Q(s(t), a(t))$ for state $s(t)$ and action $a(t)$ in iteration t.

5. The system is allowed to reach the next state, s(t + 1).
6. After observing the reinforcement signal $\mathrm{r(t+1)}$, find the value of the next state labelled $\mathrm{V_t\big(s(t+1)\big)}$ using:

$$V_t\big(s(t+1)\big) = \sum_{i=1}^{N} \alpha_i\big(s(t+1)\big).\max_k q[i,\alpha_k], \quad (3.42)$$

7. Calculate the error signal:

$$\Delta Q = r(t+1) + \gamma.V_t\big(s(t+1)\big) - Q\big(s(t),a(t)\big), \quad (3.43)$$

8. where γ indicates the discount factor and $r(t+1)$ is the reinforcement signal; $V_t\big(s(t+1)\big)$ denotes the value of the new state $Q\big(s(t),a(t)\big)$ signifying the value of the Q-function for the previous state and the action performed in that previous state.
9. The q-values are updated using an ordinary descent method described as:

$$q[i,a_i] \leftarrow q[i,a_i] + \eta.\Delta Q.\alpha_i\big(s(t)\big), \quad (3.44)$$

where η indicates the learning rate.

10. Starting with Step 2, repeat the process to determine the current state. Stop when the algorithm reaches convergence.

When the Q-Learning algorithm is finished, consequents with the highest q-values in the look-up table are used to create fuzzy rules. In summary, Algorithm 3.1 briefly describes the Fuzzy Q-Learning algorithm steps.

Finally, as stated in Chapter 2, several works where both non-fuzzy and fuzzy Q-Learning algorithms are applied in network optimization problems are available in the literature, indicating the effectiveness of combining FLCs and Q-Learning in this context.

Algorithm 3.1. Fuzzy Q-learning algorithm

Step 1: Let $t = 0$, $Q_i^0(s_i, a_i) = 0$ for all $s_i \in A$ *and* $a_i \in A$
Step 2: Select an action for each activated rule ($\varepsilon - greedy\ policy$):
$a_i = argmax_k\, q[i,k]$ *with probability* $1-\varepsilon$,
$a_i = random\,\{a_i, k = 1,2,\ldots,J\}$ *with probability* ε
Step 3: Calculate the global action:

$$a(t) = \sum_{i=1}^{N} \alpha_i(s) * (\alpha_k)$$

Step 4: Approximate the Q-function from the current q-values and the degree of the truth of the rules:
$Q(S(t), a(t)) = \sum_{i=1}^{N} \alpha_i(s) * q[i,a_i]$
Step 5: Leave the system to evolve to the next state, s(t+1).
Step 6: Observe the reinforcement signal, r(t+1), and compute the value of the new state denoted by

$$V_t(S(t+1)) = \sum_{i=1}^{N}\big(s(t+1)\big).\max_k Q[S(t),a_k]$$

Step 7: Calculate the error signal:

$\Delta Q = r(t+1) + \gamma \times Vt(s(t+1)) - Q(s(t),a)$, Where γ is a discount factor

Step 8: Update New Q-table by an ordinary gradient descent method:
$a[i,ai] \leftarrow q[i,ai] * \eta \Delta Q.\, i, \alpha i(s(t))$

Step 9: Repeat the above-described process starting from step 2 for the new current state until convergence is achieved

Some limitations of this approach are that the optimization process may be sensitive to reinforcement signal selection and the fact that the system states must be visited a sufficient number of times. However, in favour of the above described advantages, the method adopted in this thesis is based on RL.

As discussed earlier, several studies have examined the effect of using both non-fuzzy and fuzzy Q-learning algorithms when optimizing networks. These studies highlight the benefits of combining FLCs and Q-Learning. However, there are limitations as optimization processes are sensitive to reinforcement signals and the number of times system states must be visited. Regardless of the limitations, the method used in this thesis is based on RL.

One of the advantages of the reinforcement learning techniques examined in this thesis is their ability to enable a single agent to use trial-and-error interactions with the environment to learn optimal behaviours. Several RL approaches have been created that permit agents to optimize their behaviour in a variety of circumstances. Though

traditional approaches are often unsuccessful in situations where multiple learners use reinforcement learning in common environments.

3.5 Adaptation of Multi-agent based Fuzzy Reinforcement learning

Assumptions are made in multi-agent environments to assure that convergence will occur. However, these assumptions are frequently violated. Complexities are created even in simple situations where agents share a common, stationary setting and are required to only learn a strategy for a single state. In situations where the agents have opposing goals there may be no optimal solutions and establishing equilibrium between agents becomes the primary goal; essentially, agents are unable to improve their payoffs because other agents keep their actions fixed.

Dynamic environments not only have multiple agents, but they also have multiple, sequential decisions that increase their complexity. In these settings, agents must coordinate and consider the current state of their dynamic environment with very limited information. Typically, agents in dynamic environments cannot observe the actions of other agents or see what rewards they obtain as a consequence although the actions of the other agents influence their immediate environment along with the rewards they can obtain. In very complex environments agents may be unaware that other agents are present and may interpret their environment as non-stationary. Similar, equally complex environments allow agents to access information, but the state action spaces are not conducive to learning because of their complexity and the amount of coordination required between agents. Before an effective multi-agent approach can be developed, all these challenges must be addressed. A standard multi-agent reinforcement learning model is presented in Figure 3.8.

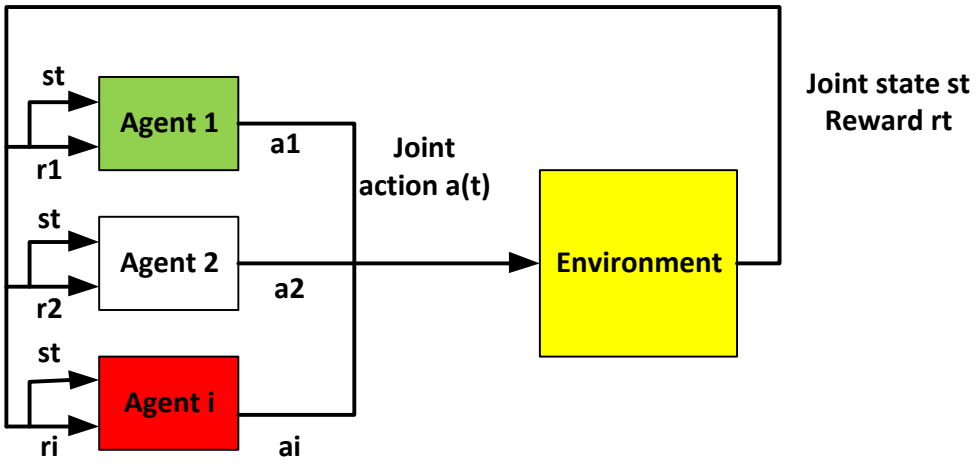

Figure 3.8: Multiple-agents acting in the same environment

Regardless of learning complexity, the demand for multi-agent systems continues to increase. In cases of systems that are decentralized and single, central learning methods are impractical. Such systems can be found when data was subjected to disruptions caused by multiple, conflicting objectives or if a single centralized controller requires too many resources. Multi-robot setups, distributed load balancing, decentralized network routing, electronic auctions, and systems designed to control traffic are all examples of such system types.

As a result of the demand for adaptive multi-agent systems and the complexity of coping with interacting learners, an increasing number of researchers have worked to develop multi-learning reinforcement methods. This field of study uses research on reinforcement learning that takes place within AI and the Game Theory. Previous studies on the Game Theory concentrated on competitive endeavours, but this field has expanded into analysing many different kinds of strategic interactions. Game Theory research has attracted the attention of psychologists, economists, biologists, the AI community, and computer scientists in general. In this thesis, the Game Theory is used to describe how attacks are detected using multi-agent fuzzy reinforcement learning techniques and approaches for analysing learning outcomes.

The focus of this thesis is on multi-agent systems that contain strategic interactions between agents. Agents are seen as autonomous entities with their own goals, the ability to make independent decisions, and that are affected by other agents' decisions. These systems are different from systems that use distributed or parallel reinforcement learning methods where multiple learners work together to accomplish a single objective. These systems can be used in advanced exploration for standard reinforcement learning and they rely on the frameworks covered by single agent theories, such as the theory described by Tsitsiklis (1994). Examples of distributed or parallel systems include methods that divide the learning state space between agents, swarm-based methods (Dorigo et al., 2010), and systems where multiple agents work together to update policies (Mariano et al., 2001). In distributed and parallel reinforcement systems, algorithm convergence is valid if any outdated information will be discarded. In other words, the max operator on the right hand side of the Q-learning update rule is permitted to use outdated Q values.

3.6 Game Theory

The game theory provides a model of strategic interactions based on individuals competing against each other in a game. A mathematical object is used to represent the game as it outlines the consequences of the interactions between players in terms of the rewards to be obtained. AI researchers often rely on extensive game forms where players take turns performing an action to model classic minimax algorithms (Russell et al., 1995). This chapter concentrates on reinforcement learning with repeated games or games in which the players simultaneously perform individual actions. In addition, the terminology and concepts used in the Game Theory are also discussed in this chapter.

Normal Form Games

Definition:

A normal form game can be expressed as $(\mathrm{n}, \mathrm{A1}, \ldots, \mathrm{n}, \mathrm{R1}, \ldots, \mathrm{n})$. In this statement, $1, \ldots, n$ represents the players, A_k signifies the finite set of individual actions available to player k, expression $R_k : A_1 \times \ldots \times A_n \rightarrow R$ states the individual reward function of player k, and his expected payoff for an action is denoted by $\boldsymbol{a} \in A1 \times \ldots \times An$.

A game begins with each player k independently choosing and individual action $\boldsymbol{a}$ from its own private set of actions $A1$. The combined actions from all players form a joint action or an action profile from a joint actin set $A = A_1 \times \ldots \times A_n$. The expected reward resulting from the joint action is described as $\in$ A, $R_k(a)$.

A payoff matrix can be used to characterize a normal form game. Examples of typical 2-player games are shown in Table 3.1. In Table 3.1, the action taken by Player 1 is represented by a row in the matrix. The actions available to Player 2 are shown in a column. The corresponding entry in the matrix identifies the payoffs Player 1 and Player 2 will receive after they complete their actions. Player 1 is sometimes called the Row Player and Player 2 is sometimes known as the Column Player. The use of more dimensional matrices in normal form player games can be demonstrated and each entry in the matrix records the payoff available to every agent once they complete a series of actions.

Strategy $\sigma_k : A_k \rightarrow [0,1]$ is an element of the probability distribution of action set A_k of player k described as $\mu(A_k)$. A pure strategy occurs when $\sigma_k(a) = 1$ for action $a \in A_k$ and 0 for any other action. If this cannot be shown, the strategy is considered mixed. A vector strategy has one strategy for each player and profile

$\sigma = (\sigma_1, \ldots, \sigma_n)$. A strategy profile will correspond to a joint action $a \in A$ if every strategy in σ is pure. One assumption made in normal form games is that the expected payoffs are linear. In other words, the expected reward in strategy profile σ for player k can be expressed as:

$$R_k(\sigma) = \sum_{a \in A} \prod_{j=1}^{n} \sigma_j(a_j)\ R_k(a) \tag{3.45}$$

where a_j represents the action available to player j in the action profile denoted a.

Game Types

The player reward function is used to classify a game. When all players hold a reward function in common, the game is classified as an identical payoff of a common interest game. A game is a zero-sum game when all the player rewards are equal to 0. In zero-sum games, some players win while others experience losses. These games are also known as purely competitive games. General sum games refer to games with no special restrictions. Examples are given in Table 3.2.

Table 3.2: Examples of 2-player, 2-action games.

	a1	a2
a1	(1,-1)	(-1,1)
a2	(-1,1)	(1,-1)

(a)

	a1	a2
a1	(5,5)	(0,10)
a2	(10,0)	(1,1)

(b)

	a1	a2
a1	(5,5)	(0,0)
a2	(0,0)	(10,10)

(c)

	a1	a2
a1	(2,1)	(0,0)
a2	(0,0)	(1,2)

(d)

From left to right: (a) Matching pennies, a purely competitive (zero-sum) game; (b) The prisoner's dilemma, a general sum game; (c) The coordination game, a common

interest (identical payoff) game; (d) Battle of the sexes, a coordination game where agents have different preferences); Pure Nash equilibria are indicated in bold.

The first game shown in Table 3.2 is also known as the matching pennies game. It is an example of a purely competitive game. In the matching pennies game either heads or tails is chosen. If both coins are the same Player 1 wins and is rewarded by Player 2. If the coins are not the same, Player 2 is the winner and is rewarded by Player 1. Evidently, a win for one player represents a loss for the other player and meets the criteria of a zero-sum game.

The second game in Table 3.2 is The Prisoner's Dilemma. It is an example of a general sum game. In this game, two criminals are held by the police in separate cells after committing a crime. Two possible actions are available to these criminals. They can deny mutual participation in the crime (action a1) or they can betray the other criminal (action a2). If both criminals take action a1, they will receive a minimal sentence (payoff 5). If one criminal implicates the other while the other continues to deny involvement in the crime, the cooperative criminal will be released (payoff 10) and his partner will be held completely responsible for the crime (payoff 0). The third possibility is that both criminals will blame the other and will be incarcerated for several years (payoff 1). In this game, the choice of blaming the other criminal dominates and could result in the best payoff. This often leads to each criminal betraying the other even though they would be better off if they had cooperated.

In the third game in Table 3.2 each player receives the same payoff if they participate in a joint action. Choosing the best joint action leads to the best payoff. A suboptimal action results in a less profitable reward and selecting the wrong cooperative action means that neither player will receive any reward. This third game is an example of a common interest game.

The Battle of the Sexes is an instance of the fourth game type illustrated in Table 3.2. Here, each player will receive their own individual rewards and each prefers different outcomes. For example, Player 1 opts for (a1,a1) whereas Player 2 prefers (a2,a2). To overcome the coordination challenge presented by their preferences, the players must reach a compromise.

Games are not always limited to two different actions. Table 3.2 shows common interest games with three possible actions. A climbing game described by Claus et al. (1998) is the first example in Table 3.2. This climbing game illustrates the Nash Equilibrium surrounded by severe penalties. In the second game, the penalties are shown on the left as parameter k < 0. The harder it is to learn the preferred solution ((a1,a1) and (a3,a3)), the smaller the value of *k* will be.

Table 3.3: Examples of 2-player, 3-action games.

	a1	**a2**	**a3**
a1	(11,11)	(-30,-30)	(0,0)
a2	(-30,-30)	(7,7)	(6,6)
a3	(0,0)	(0,0)	(5,5)

(a)

	a1	**a2**	**a3**
a1	(10,10)	(0,0)	(k,k)
a2	(0,0)	(2,2)	(0,0)
a3	(k,k)	(0,0)	(10,10)

(b)

From left to right: (a) Climbing game; (b) Penalty game, where k ≤ 0. Both games are of the common interest type. Pure Nash equilibria are indicated in bold.

Solution Concepts in Games

It can be difficult to pinpoint the desired outcome of a game because the players have different reward functions that are affected by the actions of their competitors. Players may not be able to maximize their payoffs as they may not be able to

simultaneously reach their goals. The Battle of the Sexes in Table 3.3 is a good example of this.

An important notion behind games represented by The Battle of the Sexes is the best response. A best response allows a player to capitalize on their situation in relation to the strategies used by their opponents. However, the player will not be able to increase the reward if their opponents maintain a fixed strategy. When $\sigma = (\sigma_1, \ldots, \sigma_n)$ is the strategy profile, σ_{-k} represents the same strategy profile without strategy σ_k of player *k*. The best response for strategy $\sigma_k^* \in \mu(A_K)$ of player *k* is when:

$$R_k(\sigma_{-k} \cup \sigma_k^*) \geq R_k(\sigma_{-k} \cup \sigma_k') \; \forall \sigma_k' \in \mu(A_k) \tag{3.46}$$

where $\sigma_{-k} \cup \sigma_k'$ is the strategy profile when all players use the same strategy they used in σ except for player *k* who uses σ_k', i.e. $(\sigma_1, \ldots, \sigma_{k-1}, \sigma_k', \sigma_{k+1}, \ldots, \sigma_n)$.

The Nash equilibrium mentioned above is an instance of a central solution. When using the Nash equilibrium, all players act on their mutual, best replies. Every normal form game has a minimum of one Nash equilibrium (Nash, 1950). The Nash equilibrium for each player can be expressed as a strategy profile of $(\sigma_1, \ldots, \sigma_n)$. Strategy σ_k represents the best response to the strategies a player's opponents signified by σ_{-k}. No player can enhance their reward or payoff if they deviate from playing the Nash equilibrium. As a result, a single player has no motivation to independently change their strategy. The only way to escape the Nash equilibrium is for several players to change their strategies simultaneously.

Definition. A strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ is called the Nash equilibrium if for each player *k*, strategy σ_k is the best response to the strategies of the other players σ_{-k}.

Thus, when playing the Nash equilibrium, no player in the game can improve their payoff by unilaterally deviating from the equilibrium strategy profile. As such, no

player has an incentive to change their strategy, and multiple players must change their strategies simultaneously in order to escape the Nash equilibrium.

Justification of the selected technique

Amongst the techniques explained in the literature review, the game theory-based RL was selected in this thesis. The main reasons for discarding the other alternatives as well as for choosing a game-based RL method is discussed subsequently.

First, although Neural Networks have been successfully applied in many applications, this artificial intelligence technique has some limitations and disadvantages. On the one hand, neural networks are especially appropriate for prediction, function approximation, classification, pattern recognition, and clustering, which are not tackled in this thesis, and they mainly focus on developing control techniques. An important drawback is that neural networks require a large diversity of training for real-world operation, which can be a severe constraint in complex systems such as real-time traffic monitoring in networks. In addition, neural networks cannot be trained a second time, in the sense that it is very hard to add new data to an existing network. Finally, they require abundant computational resources and high processing time for large neural networks.

Secondly, although genetic algorithms are easily understood and transferred to current simulations and models and do not require advanced mathematical skills, they also have several important limitations and disadvantages including:

- There is no solid guarantee that a global optimum will be found by a genetic algorithm. Global optimums typically occur with larger populations.
- Genetic algorithms involve the problem of genetic drift, which is a major problem of genetic algorithms. This means that the genetic algorithm may quickly lose most of its genetic diversity and the search then occurs in a way that

is not beneficial for recombination. This is because the random initial population rapidly converges.

- Genetic algorithms face limitations with control problems that are executed in real-time due to convergence and random solutions. This indicates improvements have been made in the entire population but those improvements cannot be extrapolated to an individual within the population. As a result, it is ineffective and impractical not to test genetic algorithms on simulation models prior to using them in on-line control in real systems.

It is worth mentioning that in the context of this thesis, network operators may be reluctant to implement this kind of algorithms since the solutions found by genetic algorithms can remain at a certain distance from the optimum with higher probability, leading to suboptimal performance in an undetermined amount of time. In addition, the slow convergence of this technique can also be an issue in real systems, such as networks, even if an off-line control is applied.

Third, in particle swarm optimization, there are some important limitations related to the optimization of an FLC in the context of networks (Gupta et al., 2005). In particular, particle swarm optimization involves loss of information in the global cost functions, since performance indicators are globally measured in the concerned network area. Thus, the situation at sensor nodes cannot be considered. In addition, this optimization method has to compare the evolution of many particles, each of which represents a different FLC setting. As a result, to assess the position of each particle, the corresponding FLC should be evaluated in many sensors, thus requiring exclusive use of simulation tools. The lack of flexibility and generality in defining FLC is also a constraint for particle swarm optimization. In this sense, when adding new inputs or performance indicators to the FLC, the optimization phase must be launched, although

certain a priori knowledge can be included at the beginning. Other disadvantages of the basic particle swarm optimization algorithm are covered by Gaing (2004) and Shi (2001). These disadvantages of the algorithm include slow convergence during the refined search stage, ineffective local search capabilities, and possible entrapment in the local minimum. Currently, there are no mathematical proofs of the convergence and speed of convergence for this algorithm.

Finally, in Game-based IDPS, the combination of FLCs and fuzzy Q-Learning algorithm is highlighted as a powerful mechanism in the context of networks for the following reasons:

- Attacks in networks are complex and variable systems, in which obtaining a training data set that is representative of all situations becomes a difficult task. Unlike other approaches (e.g. supervised learning in Neural Networks), Fuzzy Q-learning does not require a training data set.
- Due to the complexity of network management, operators do not usually have the knowledge necessary (i.e. accurate and complete) to take proper action in every attack situation. In this case, learning from interaction becomes a suitable solution, where a multi agent is able to learn from its own experience to perform the best actions.
- It is possible to perform the optimization at a distributed level, so that many FQL agents can learn in parallel. To achieve this, measurements should be taken in the area of a network. In addition, the cooperative-based fuzzy Q-Learning method provides operators generality to easily introduce, for instance new performance indicators in IDPS.

3.7 Collaborative-IIDPS Architecture

To design a Collaborative IIDPS (Co-IIDPS) based on a comprehensive set of requirements for networks, the special characteristics of a distributed framework structure are scrutinized in this thesis, within Smart Grid networks with Collaborative IIDPS as proposed by Patel et al. (2013). Figure 3.9 shows the combination of Network and Host-based IDPS (NIDPS, HIDPS) in a fully distributed framework structure in a Smart Grid networking environment with Collaborative-IDPS. This formation is readily applicable to any network.

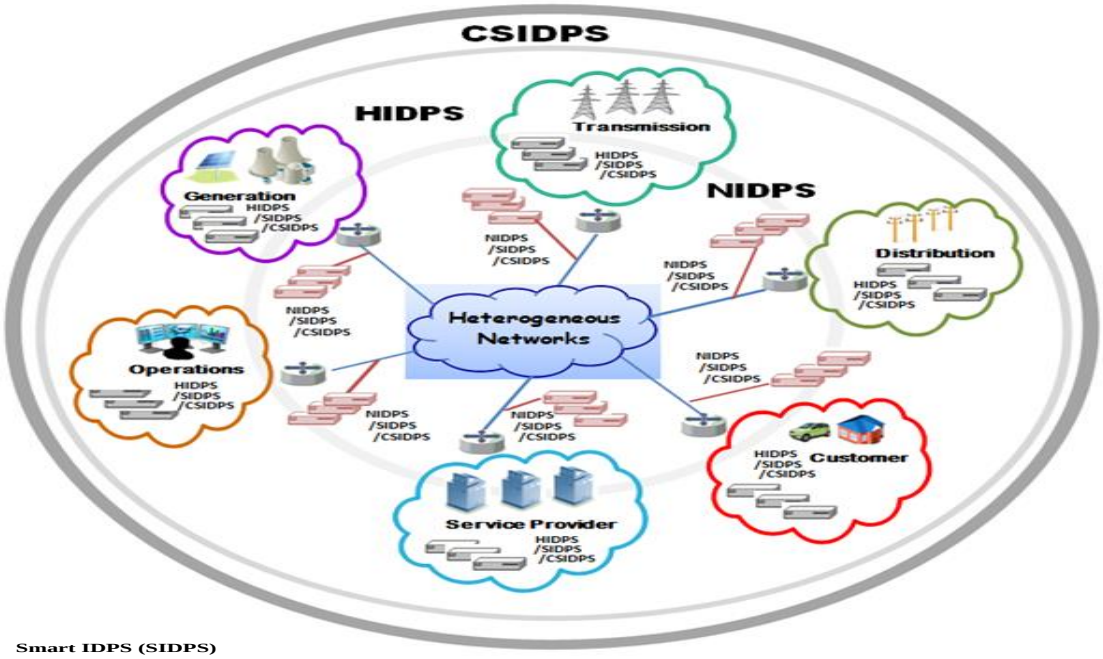

Figure 3.9: Combination of NIDPS and HIDPS in a distributed Smart Grid Network (CIDPS)

The monitored environment of an IDPS is typically specified as:

- A network-based IDPS monitors network traffic for particular network segments or devices, and analyses the network and protocol behaviour to identify suspicious activities.
- A host-based IDPS monitors all or parts of the dynamic behaviour and state of a computer system. Unlike NIDPS, which dynamically inspects network

packets, HIDPS detects which programs access what resources. HIDPS has the advantage of being easier to deploy without affecting existing infrastructure compared to NIDS, which detects attacks at the transport protocol layer via quick responses. However, a combination of both HIDPS and NIDPS solves the problem of assimilation and scalability through collaborative management.

Due to the IIDPS complexity in a network, this chapter incorporates three newly defined concepts of detection management: Fuzzy Reinforcement Learning Management (FRLM), Knowledge Management (KM), and Multi-agent Management (MA) into the core architectural design of CIDPS (Figure 3.9). Management flows from the module of computational intelligence intrusion detection through an intermediate section are viewed as a fuzzy reinforcement learner, and knowledge and multi-agent managers, and are expected to respond to intrusions in WSN. The correlation flows are developed according to the collaborative-IDPS and desired IIDPS characteristics. The purpose of the thesis encompasses three concepts, namely fuzzy system, reinforcement learning and a multi-agent system. They are intended to design an efficient system that meets the Collaborative-IIDPS (Co-IIDPS) requirements.

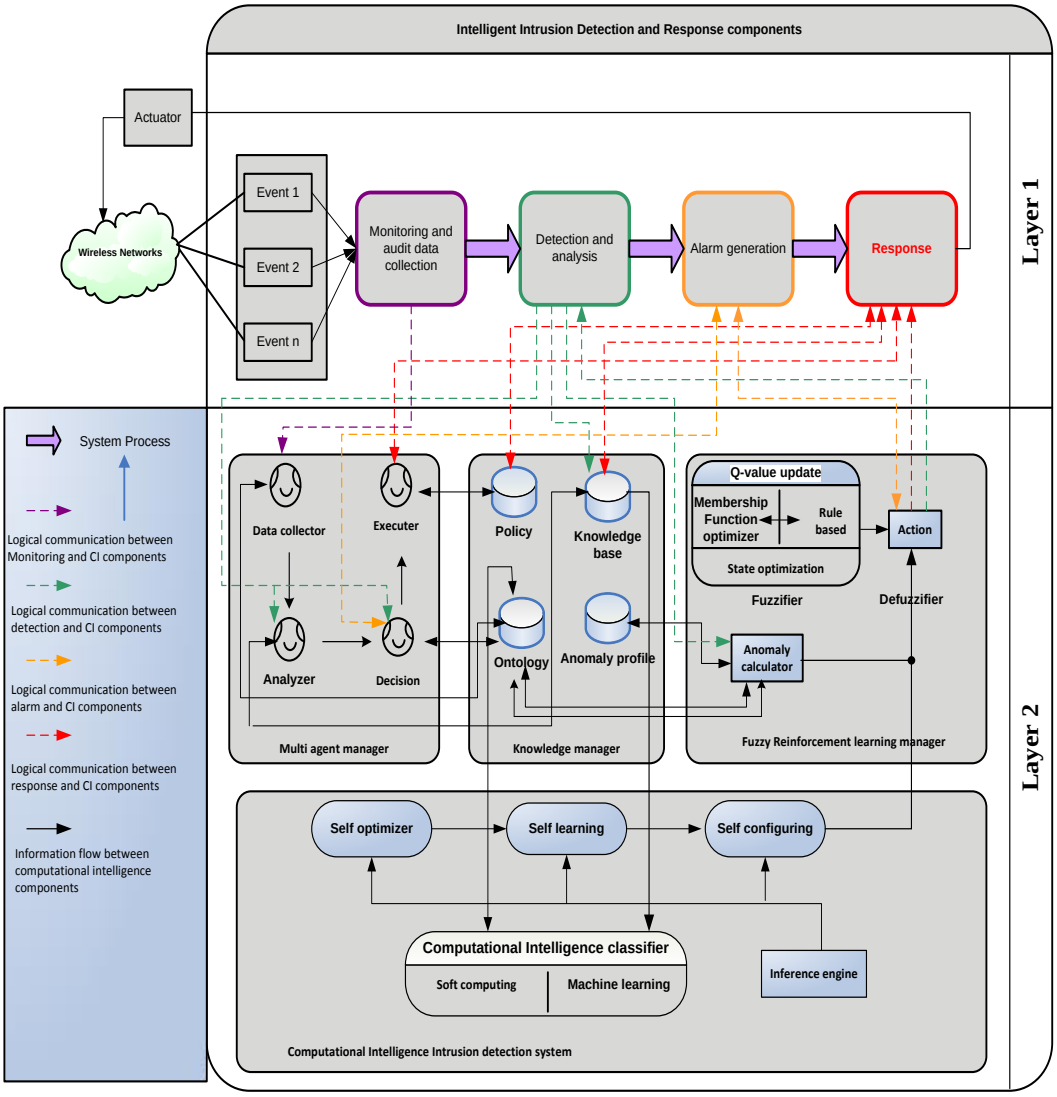

Figure 3.10: Enhanced Collaborative-IIDPS functionality architecture within a network

Figure 3.10 shows the Co-IIDPS functions. The first layer shows the traditional system components that monitor and collect the audit data through the sensors, analyse the data and detect intrusions, generate alarms and herald the proper response through the actuators. The advanced components seen in layer two are drawn from the four proposed concepts.

The advanced components employ computational intelligence (CI) techniques such as soft computing (e.g. neuro-fuzzy systems) and machine learning (e.g. a reinforcement learning system) to detect intrusions and feed the obtained results into the autonomic solution mode components comprising a self-optimizer, self-learning, and self-configuration. Self-learning and self-optimizing are defined in autonomic

computing principles in real-time without human intervention. The proposed collaborative IIDPS system architecture is illustrated and presented as a workflow scenario to show the way it functions in 8 steps as numbered in brackets. The arrows point out the information flow between components while the dash arrows indicate the logical communications between the components.

Inputs from Autonomic Wireless Environment Components:

The processor, memory, RF radio, power source, and actuators constitute the wireless network components. The interactions between them generate and prepare the input sensor signals from the wireless environment. The signals together with the events or latest challenges pass through the intelligent intrusion detection and prevention system (IIDPS) components for analysis. The monitoring, detection, alarm generation and response utilize computational intelligence methods to mitigate IDPS.

Latest IIDPS Challenges & Enterprise IIDPS Policies:

The Intrusion Detection and Response System (IDRS) Policies and Latest Challenges to networks fall into the computational intelligence intrusion detection module of autonomic network -based IDPS, as mentioned in Step 1. An event entering the system is checked to determine whether it is an intrusion. If it is an intrusion, the Intrusion Detection Engine (IDE) takes full responsibility for analysing and identifying the type of attack.

Computational Intelligence Intrusion Detection Systems (CIIDS):

Various CIIDS techniques have been suggested in this architecture. Machine learning and soft computing are the main CIIDS methods. Reinforcement learning (RL) together with fuzzy sets (FS) serves as a feature extraction selector and classifier of machine learning for Co-IIDPS. The results of signal classification for intrusion detection are relayed to the inference engine.

Inference Engine (IE):

The IE is a logical, key division of the Co-IIDPS. The IE performs based on the latest computational intelligence techniques, fronted and equipped with fuzzy reinforcement learning management.

Fuzzy Reinforcement Learning Management:

The fuzzy reinforcement learning (FRL) management (FRLM) of this Co-IIDPS architecture includes reinforcement learning (RL) and fuzzy sets (FS). Given an anomaly incident, FRLM is internally analysed and it updates the Q-value of the learner's agent. If necessary, it automatically updates a newly discovered intrusion incident by applying anomaly calculator component-based computational intelligence and knowledge management techniques in recursive iteration of its execution cycle.

Knowledge Management:

In order to share knowledge and allow collaboration between other managers (i.e. Fuzzy Reinforcement Learning and Multi-agent managers), the knowledge manager (KM) uses four types of decision mechanism: policy, ontology, anomaly profile and knowledge-based. The knowledge-based component directly connects to CIDPS to store the process of training and testing CI algorithms. The purpose of decision ontology (DO) is to provide a basis for representing, anomaly modelling and probing the decision to identify abnormal behaviour. The policy works as an action selector and uses an executer agent. The action policy of the KM mechanism adapts to FRL to cluster the incidents according to severity and raise an alarm.

Multi-agent Management:

A multi-agent manager prioritizes an anomaly according to a victim's vulnerability. There are three possible scenarios in this state. The first case is pattern

collection through the data collector agent (DC). In the second case, if the incident is a severe intrusion or low intrusion, the analyser agent (AA) and decision agent (DA) are associated to the ontology and knowledge base, which comprise the computational intelligence techniques for intrusion identification. The third possible situation is related to the executor agent (EA) that is shielded from intrusions before any data loss or damage happens. It provides the impetus for the system to self-learn against any attacks, as well simultaneously purvey for protection and prevention capabilities further down the chain in the autonomic mode of operation.

Self-optimizing, Self-learning, and Self-response:

The second case in Step 7 is indicative of some parts having already been attacked or even infected. In the case of action selection, the penetration tracks in Co-IIDPS activate the self-optimizing component to ensure the system protects itself. The third situation refers to intrusion blocked prior to any data loss taking place. Here, the system automatically enters a self-learning state. In both circumstances, the self-optimizing state is triggered directly after Knowledge Management and self-learning are performed after Fuzzy Reinforcement Learning Management to protect the system by either computational intelligence fuzzy methods or reinforcement learning, or a combination. These methods are triggered to protect the system by updating the Co-IIDPS as a whole. Their actions are defined by the Inference Engine component in Co-IIDPS. Signals are then sent to activate actuators to execute prevention in the network environment.

3.8 Discussion

The discussion is expanded with the proposal of a new architecture to detect and prevent intrusions in a network, by combining computational intelligence and multi-agent based computational intelligence approaches. A novel collaborative-based IIDPS

(Co-IIDPS) architecture was proposed and presented. It demonstrates the impact of a Multi-agent system-based computational intelligence (MCI) technique on enhancing detection efficiency and false alarm rates. This architecture portrays the clear notion of cooperative learning-based detection to satisfy the requirements of IIDPS. The projected architecture defines three detection means of management: Fuzzy Reinforcement Learning Management, Knowledge Management, and Multi-agent Management. In conclusion, the detection management techniques can be improved by minimizing the false alarm rates and increasing the detection rates in addition to decreasing energy consumption in networks. In the next chapter, with the intent of validating the proposed architecture, the aim is to design and develop the aforementioned detection management components using game theoretic approaches.

3.9 Chapter Summary

In the context of Co-IIDPS, adaptive game theoretic techniques are adequate for network parameter optimization due to network complexity and dynamism. The main benefits of applying such techniques are cost savings and improved network performance. This chapter began with a description of potential network self-tuning approaches. In this analysis, the use of a network model was discarded, since constructing a network model that is accurate and manageable is usually a complex task that may lead to poor performance as well.

Thus, the schemes adopted in this thesis are based on self-tuning entities that interact directly with the network. In such cases, a closed-loop structure is used to find the optimal parameter settings. For this reason, the next part of the chapter focused on controllers based on the Fuzzy Logic theory, as this discipline provides a mathematical framework especially appropriate for designing controllers. Its potential lies in the capability to express knowledge in a similar way to human perception and reasoning.

The second topic in this chapter was devoted to mathematical approaches that can be used to optimize and adapt the behaviour of FLCs. The first technique is Neural Networks, which typically requires a training data set that can be a severe constraint in networks. In addition, neural networks are more suitable for other kinds of problems (e.g. prediction, classification, pattern recognition, and clustering), which differ from the control problem addressed in this thesis. Secondly, the main feature of Genetic Algorithms was described, showing that their application in control problems performed in real-time is limited owing to random solutions and convergence. Third, the basic concepts of the Particle Swarm approach were presented, highlighting its application to FLC optimization involving some important limitations, such as lack of flexibility and generality in defining the FLC. Finally, this chapter was devoted to RL, which is the method selected amongst the previously described ones. The main benefit of this approach is that RL algorithms learn from interaction, which becomes essential in complex systems such as networks.

The third topic in this chapter dealt with the proposal of game theoretic approaches that can be used to optimize and adapt the behaviour of Fuzzy Q-learning. The reinforcement learning manager emerged as a result of applying fuzzy techniques to Co-IIDPS, leading to robust, fault-tolerant and easy to manage and operate WN architectures and deployments. Knowledge management enables the characterization of anomaly profile knowledge as a set of related concepts within an anomaly calculator domain. The policy aspect of a multi-agent manager is thus utilized to predict anomaly behaviour. In summary, the scalable, fully distributed structure of our system exposes the risks of low accuracy detection and difficulty in synchronizing information between autonomous agents.

Chapter 4 : GAME THEORETIC APPROACH USING FUZZY Q-LEARNING

The novelty of this study lies in the proposal of a game theoretic framework, namely the cooperative Game-based Fuzzy Q-learning (G-FQL) in order to identify attackers and appropriately respond to them. The aim is to facilitate an intelligent intrusion detection and response mode. Thus, the current evaluation study is significant in that the feasibility and suitability of the framework are highlighted.

In this chapter, the game framework design is first explained in three sub-sections: the player strategies, the player payoff function, and an analysis of the reward function. In addition, a utility function is employed to evaluate the effectiveness and performance of the model. A detailed explanation of the fuzzy Q-learning adapted to the game theory is also given.

The game theory is a branch of applied mathematics that deals with the way rational entities or agents make decisions in the application of WSNs (Huang et al., 2013), cognitive radio networks (Elias et al., 2011), and ad hoc networks (Naserian et al., 2009). It affords an array of mathematics tools for modelling and analysing the interactions among rational groups, whereby rationalism is founded on the profit or reward perceived by the entities (Shoham et al., 2009). An anomaly-based wireless network in the game-theoretic approach is a tremendously difficult task on account of the distributed nature of numerous players in WSNs. A large number of players additionally results in difficulty achieving equilibrium in a competitive game. To deal with a certain type of attack in wireless networks, Naserian et al. (2009) included an assortment of games, for instance non-cooperative, two-player, and non-zero-sum to their stratagem. In such game arrangements, better decisions are made according to the principles offered by payoff functions. Shen et al. (2011) took into account the signalling game to create an IDPS game that exhibits the interaction between an attacker

and cluster head in a WSN. The Bayesian Nash Equilibrium (BNE) scheme in conjunction with the mixed strategy for outstanding detection policies served as the basis for their model. Thus, an ideal, fundamental shield tactic to protect WSNs was achieved, while the probability of detecting attacks was simultaneously, considerably enhanced.

A multi-agent system utilizes the reputation security mechanism to perform dynamic role assignment based on the following three parameters: reputation, bootstrap time and energy. The approach evicts highly non-cooperative and malicious nodes from the network (Misra et al., 2011). An adaptive learning routing protocol employs a learning automata algorithm for efficient malicious node detection (Rolla et al., 2013). The multilayer reinforcement learning framework assisted by the Hidden Markov Model (HMM) was proposed to solve real-time detection in a complex state space (Andersen et al., 2009). The results indicated that the network's cost function could be optimized if the agents collaborated repeatedly. In our proposed scheme, the cooperative game is implemented into IDPS to generate the benefits of a fuzzy Q-learning algorithm with a value function to mitigate the flooding attack issue in a WSN with respect to detection and defence accuracy. Resource loss, accuracy of attack detection via sensors, and service inaccessibility at critical times are among the challenges posed, and in this thesis, an effort is made to confront the security setbacks by applying the cooperative game-based fuzzy system and reinforcement learning mechanism.

4.1 Proposed model

4.1.1 WSN model

In the present research study, Figure 4.1 illustrates the distributed network with hierarchical routing, which consists of Clusters (C), their coordinators, or Cluster Heads

(CHs), as well as the member Sensor Nodes (S). In the current scheme, the Cluster Head (CH) is assumed to be a Sink Node (SN) in each cluster. The SN monitors the behaviour of sensor nodes by collecting data from the member sensor nodes and transmitting the critical status -- the attack information of the sensor nodes, to a Base Station (BS). Each cluster is mapped into distributed system formation while the set of sensor nodes is mapped into each cluster grouping. Although only one BS is shown in Figure 4.1, there could practically be several implemented in a real operational WSN.

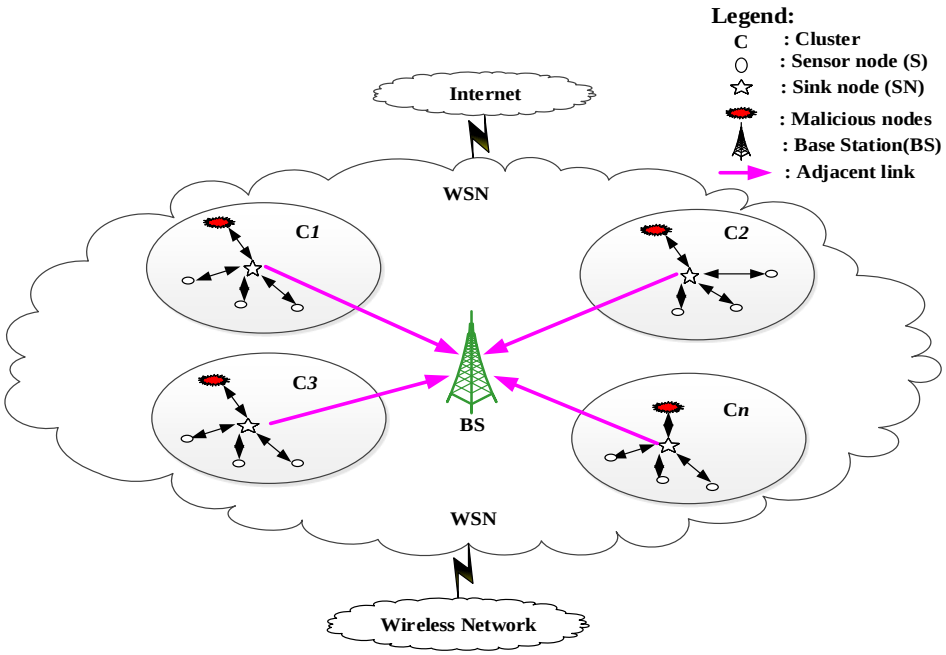

Figure 4.1:A distributed hierarchical system perspective of a WSN

The route from a sensor node (S) to a base station (BS) is deemed a distributed-hierarchical path that creates a hierarchical system with numerous routes, which is the main feature of cluster-based WSNs. Sensor nodes function independently to avoid the collapse of all sensor nodes (SNs) in case one fails. The sensor node redundancy approach increases the overall reliability in distributed hierarchical systems. Figure 4.1 illustrates how SNs send data gathered from a sink node to a BS via other adjacent SNs, and the BS receives data only if all SNs within the routing formation are actively functioning. Hence, a set of clusters on a route is counted as a set of independent distributed-connected elements. Attacks in this scenario can target the WSN in multiple

ways, with DDoS attacks potentially originating either from the Internet or neighbouring wireless sensor sources.

4.1.2 Methodologies and Techniques Used

Game-based detection and the defence mechanism operate to detect DDoS attacks, where the sink node and base station adapt to select the best strategy of detecting an immediate attack and responding to it. Regardless of whether attacks are carried out on a regular or irregular basis, the IDPS can adjust its learning parameters through fuzzy Q-learning to identify future attacks. The architecture of the proposed game-based FQL is dual, in that it has two phases (Figure 4.2).

Phase 1: In the primary game scenario stage, player 1 (the sink node) utilizes the fuzzy Q-Learning algorithm to identify the level of disruption caused by the attacking player to the victim node, leading to anomalies such as low access or damage. For attacker player detection, the sink player adopts three strategies: catch, missed, and low catch, as elaborated in player strategies applied to the sink node. Finally, the sink node transmits an alarm event that contains malicious node information to the base station (player 3) via an adjacent link connected to the base station (Figure 4.2). The malicious information is pre-processed by the sink node to travel from phase 1 to 2 based on the alarm event beyond the default value threshold, in order to prepare a countermeasure strategy against the attacker through a defence mechanism.

Phase 2: In the second phase of the game scenario, player 2 (the base station) employs the fuzzy Q-learning algorithm to confirm the malicious node's behaviour. It checks the memory of player 1 or looks it up in a table and compares it with its memory in order to defend against the attacker. The detection player (sink node) and defence player (base station) coordinate their defence with each other to shield the wireless sensor nodes against the attacker player (attack/intrusion).

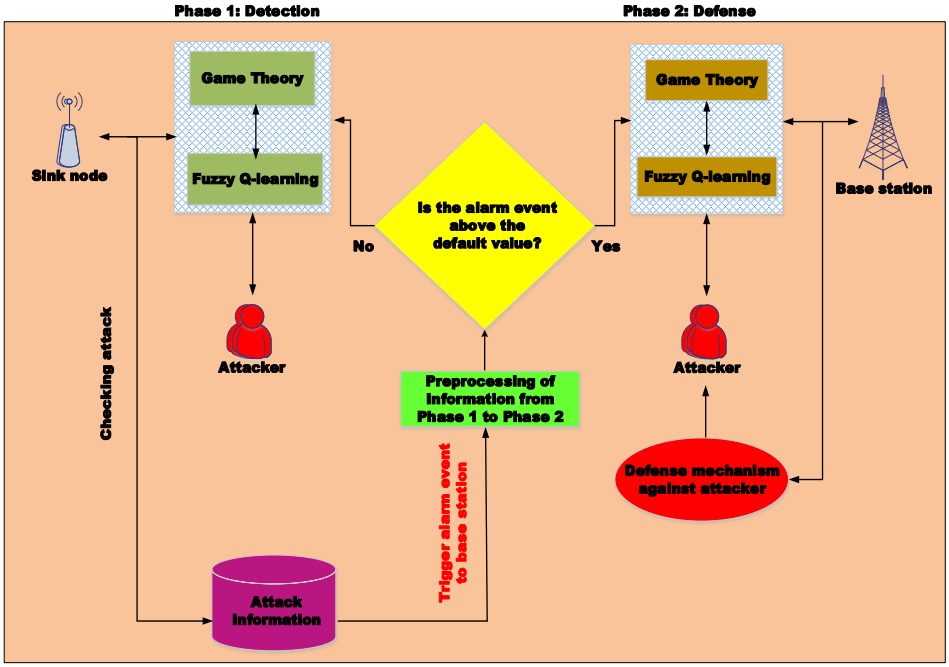

Figure 4.2: Model of a Cooperative Game-based IDPS and an attacker

To highlight the proposed game-based FQL, the sink node and the base station are allocated a corresponding reward/incentive functional value, which is retained by the Fuzzy Q-learning IDPS. As such, a node's evolving fuzzy state may be recorded and quantified through the fuzzy reward utility function as discussed in the player payoff function. When a node encounters an attack or receives an anonymous message, the sink node sends the related severity alarm event evidence and messages to the BS, who then analyses the critical data to adjust the FQL parameters. Based on the sink node information, the base station decides which nodes are under attack or at risk and elects whether to safeguard them or not. The BS previously set a severity alarm event threshold rate, v. Once the severity alarm value acquired by a node exceeds v, the FQL IDPS deems the node under attack or at risk and strengthens its defences to secure the cluster area in which the node is detected at the associated base station.

4.1.3 Possible attack categories

In this thesis, the Open System Interconnect (OSI) model is classified into five layers (Akyildiz et al., 2002): Physical layer, Link/MAC layer, Network layer, Transport layer, and Application layer. The attacks in each layer are analysed by focusing on the flooding attack and its potential defences. In the proposed scheme, a specific kind of DDoS attack is created with respect to a flooding attack that affects cluster heads. The generated attack sends flooding UDP packets to diminish the cluster head's energy.

Table 4.1: Classification of Denial-of-Service attacks and defence at each protocol layer

Protocol Layer	**Attack**	**Defense Mechanism**
Application Layer	Overwhelming (McGregory, 2013)	Sensor tuning
		Data aggregation
	Path-based DoS (B. Li et al., 2009)	Authentication and anti-replay protection
	Deluge (reprogramming) attack	Authentication and anti-replay protection
		Authentication streams
Transport Layer	SYN (synchronize) flood (Bicakci et al., 2009)	SYN cookies
	De-synchronization attack (Xing et al., 2010)	Packet authentication
Network Layer	Spoofing, replaying, or altering routing control traffic or clustering message (Qazi et al., 2013)	"Authentication and anti-replay protection secure cluster formation"
	Hello floods (Khalil et al., 2010)	"Pairwise authentication"
		"Geographic routing"
	Homing, black-hole attack (Khalil et al., 2012)	Header encryption
		Dummy packets
Link/MAC (medium access control)	Jamming (Law et al., 2005)	Authentication and anti-replay protection
	Denial of sleep (Law et al., 2009)	Authentication and anti-replay protection
		Detect and sleep
		Broadcast attack protection
Physical Layer	Jamming (Z. Li et al., 2012)	Detect and sleep
		Route around jammed regions
	Node tampering or destruction (Xing et al., 2010)	Hide or camouflage nodes
		Tamper-proof packaging

Table 4.1 indicates the impact of such attacks on WSN layers as well as the defence mechanism. In this thesis, a type of DDoS attack is considered. It is characterized by the presence of an attacker, and is known as a UDP flooding attack. In the proposed model, a UDP flooding attack occurs based on a random function to compromise the CH in each cluster. This kind of DDoS attack is aimed at exhausting CH energy by sending flooding packets in a fraction of time (Ghosal et al., 2013).

4.2 The architecture of cooperative game-based FQL IDPS

The proposed game-based defence strategy is primarily a combination of the cooperative game theory and fuzzy Q-learning algorithm. The game-based detection and defence mechanism work to detect DDoS attacks, where the sink node and base station adapt to select the ideal strategy of detecting an immediate attack and respond to it. Regardless of whether the attacks are carried out on a regular or irregular basis, the IDPS can adjust its learning parameters through fuzzy Q-learning to identify future attacks. A comprehensive description of the theoretical and practical operation of the game theory and Q-learning modes, mainly concerning Fuzzy Q-learning, is provided later. Cooperative game-based architecture in a wireless network is proposed as well (Figure 4.3).

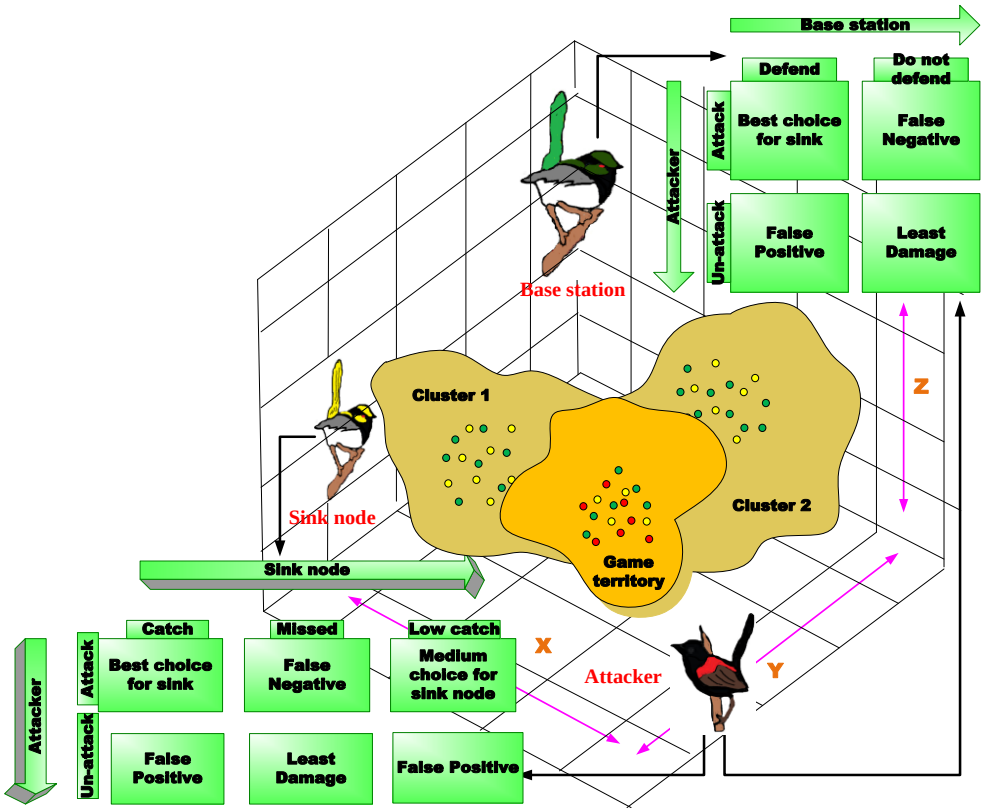

Figure 4.3: Game-based defense system architecture

In the primary stage of the game scenario, player 1 (the sink node) utilizes the FQL algorithm to evaluate the contents of the attacker player's level of access (i.e. low access, or damage). With regard to detection, the sink node player assumes three strategies, namely catch, missed or low catch. Upon completing the first stage, the sink node transmits an alarm to the base station (player 3) when the attacker assaults the sensor node. In the second phase of the game scenario, player 3 (the base station) employs the FQL algorithm to evaluate the attack records in order to defend against the attacker. The detection player (the sink node) and defence player (the base station) participate in a game via a 3D game interface to shield the wireless sensor nodes against the attacker player (the attack).

4.2.1 Game Design

In the proposed game theory method, it is assumed that the sink node can identify abnormalities in view of IDS1. Accordingly, in computer-generated WSNs, the

sink player or cluster head diffuses the alarm to the base station (IDS2) upon perceiving an anomaly. When the IDS1 receives an anomaly message, it acquaints itself with this sort of attack using the FQL detection algorithm and archives the information in its attack record database. The IDS2 attempts to respond to these attack records. The fundamental concepts of the proposed game theory, player strategies and player payoff function are introduced next.

Player strategies

The interactions between the G-FQL and attackers are split into two main categories (Tables 4.2 and 4.3 respectively). The first category represents a game between an attacker and sink node players, while the second type denotes a game between an attacker and base station player. The game play strategy between a sink node and an attacker with respect to IDS1 comprises:

1) *Best choice for sink:* The sink node chooses to identify the attacker, and the invader opts to attack.
2) *False negative:* The sink node chooses not to identify the attacker, and the attacker strikes.
3) *Medium choice:* The sink node chooses to identify the attacker with *low catch*, and the attacker attacks.
4) *False positive:* The sink node elects to detect the attacker, and the attacker chooses not to attack.
5) *Least damage:* The sink node chooses not to identify the attacker, and the attacker chooses not to attack.
6) *False positive:* The sink node chooses to identify the attacker with *low catch*, and the attacker chooses not to attack.

Table 4.2: Game play between a sink node (IDS1) and an attacker

Game play between sink node and attacker		**Sink**		
		Catch	**Missed**	**Low Catch**
Attacker	**Attack**	(a11,b11) = Best choice for sink	(a12,b12) = False Negative	(a13,b13) = Medium choice for sink node
	No attack	(a21,b21) = False Positive	(a22,b22) = Least Damage	(a23,b23) = False positive

The game strategy between a base station and an attacker concerning IDS2 is defined as:

1) *Best choice for base station:* BS elects to defend and the attacker decides to attack;
2) *False positive:* BS elects to defend, and the attacker chooses not to attack;
3) *False negative:* BS elects not to defend, and the attacker attacks;
4) *Least damage:* BS elects not to defend, and the attacker chooses not to attack.

Table 4.3: Game play between a base station (IDS2) and an attacker

Game play between base station and attacker		**Base station**	
		Defend	**Do not defend**
Attacker	**Attack**	(a11,c11) = Best choice for sink	(a12,c12) = False Negative
	No attack	(a21,c21) = False Positive	(a22,c22) = Least Damage

The player payoff function

In this thesis, a payoff value is defined as a player reward function if it protects the WSN. In other words, when the IDPS fails to defend the WSN in case an invader attacks, the player's payoff would be different. The three player payoffs are expressed as A, B, and C, where a_{ij}, b_{ij}, and c_{ij} denote the sink node, attack and base station payoff, respectively. Table 4 displays the payoff matrix, utility function as well as a description of the utility function.

Table 4.4: The payoff matrix and utility functions

Payoff function	**Payoff matrix**	**Utility function**	**Description of Utility function**
Attacker's payoff function	$A = [a_{ij}]_{2*3}$	$a_{ij} = IR - Cost_{processing}$	$IR = \frac{Number\ of\ malicious\ attacks}{Total\ malicious\ attacks\ sent}$
			$Cost_{processing} = processing\ time\ for\ attack$
Sink Node's payoff function	$B = [b_{ij}]_{2*3}$	$b_{ij} = P_d - Cost_{process\ detect}$	$P_d = \left(\frac{Correct\ attack\ detection}{Total\ detection\ and\ no\ detection}\right)$
			$Cost_{process\ detect} = Cost\ of\ attack\ detection\ during\ sink's\ processing$
Base station's payoff function	$C = [c_{ij}]_{2*2}$	$C_{ij} = P_k - Cost_{defend}$	$P_k = Cost\ of\ killing\ attacks$
			$Cost_{defend} = power\ cost\ during\ defense\ against\ attack$

Attacker's payoff function

The attacker's payoff matrix $A = [a_{ij}]_{2*3}$ is defined as follows:

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}_{i*j}$$

where $a_{11} = IR - Cost_{processing}$ represents $\left(\frac{\text{Number of malicious attacks}}{\text{Total malicious attacks sent}}\right) -$ (processing time for attack), which is when an attacker and the sink node choose the same sensor nodes to attack and detect, respectively (AS1, SS1). The attacker's original utility value of U(t) will be deducted from the cost of attacks. $a_{12} = IR - Cost_{processing}$ represents an instance when the attacker attacks and the sink node does not detect it correctly. However, $a_{13} = IR - Cost_{processing}$, means that an attacker hits and the sink node detects a compromised node with a low rate of detection. $a_{21} = Cost_{processing,}$ signifies that an attacker does not attack at all, but the sink node falsely detects the attacker. By subtracting $IR = \left(\frac{\text{Number of malicious attacks}}{\text{Total malicious attacks sent}}\right)$ from the original utility function, $a_{22} = Cost_{processing}$ signifies that the attacker and sink node choose two different strategies, neither of which causes an attack nor detects an attack correctly, respectively. In this case, the cost of attacking one node from the original utility is ignored. When $a_{23} = Cost_{processing}$, it signifies that the attacker does not attack and the sink node detects the attack with low probability/performance.

Sink node payoff function

By denoting the sink node's payoff with matrix $B = [b_{ij}]_{2*3}$ we get:

$$B_{ij} = \begin{bmatrix} b_{11} = P_d - Cost_{process\ detect} & b_{12} = Cost_{process\ detect} & b_{13} = P_d - Cost_{process\ detect} \\ b_{21} = P_d - Cost_{process\ detect} & b_{22} = Cost_{process\ detect} & b_{23} = P_d - Cost_{process\ detect} \end{bmatrix}_{i*j}$$

where:

$$P_d = \left(\frac{Correct\ attack\ detection}{Total\ detection\ and\ not\ detection}\right)$$

$Cost_{process\ detect}$: is the cost of attack detection during sink processing

Base station payoff function

By describing the base station's payoff function with matrix C= $[c_{ij}]_{2*2}$, it is defined as: C_{ij}= $\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}_{2*2}$

where $C_{11} = P_k - Cost_{defend}$ denotes $(Cost\ of killing\ attacks) - (the\ cost\ of\ defending\ against\ attack)$, which is when a base station and attacker opt for the same sensor nodes to attack and defend, respectively.

Reward function analysis

According to a three-player game, two constant reward values are defined as: R1 for the gain of the IDS1 when the sink node identifies the WSN, and reward value R2, or positive reward, as the gain of the IDS2 when the base station protects the WSN. If the sink node does not identify the WSN during the attack, the IDS1 reward would be –R1 (a negative reward). Likewise, if the base station fails to defend the WSN during an attack, the IDS2 payoff would be –R2. An explanation of the correlated reward/incentive functions of a sensor node and base station is provided in Table 4.5. To detect a potential, future DDoS attack on a sensor node, Fuzzy Q-learning is applied to enhance the self-learning ability of the IDS1 and IDS2 processes. The Fuzzy Q-learning supplies the IDPS with a learning mechanism, but the self-learning ability of the Q-learning IDS can evolve during the learning process, something that takes learning time, especially at the beginning. Through such self-iterative learning, IDSs are capable of protecting sensor nodes from recognizable potential attacks in ongoing, active WSNs.

Table 4.5: Notations associated with the reward functions of a sink node and base station

T	T = {0, 1, …, k −1} denotes the set of time in a Markov process
S	The fuzzy state space of a sensor node, where the initial state is S0, and the next state of si is si+1 for I ∈ T
D1, D2	The set of detection strategies
-R1,-R2	The payoff incurred at a false negative incident

Fuzzy Q-learning is a discrete-time fuzzy-based Markovian procedure. When the process is at time *t* and fuzzy state *FSt*, the Decision Maker may choose to perform a fuzzy action. The process responds with a corresponding fuzzy reward for the decision maker at time *(t+1)* and moves to fuzzy state *Fst+1*. The interaction details and information are as follows. Based on the Fuzzy Q-learning concept, a function fx(1): FS1 → FD1 × FA1 is defined to demonstrate the detection and attack strategies for node *x* at a specific interval in IDS1. For instance, fx(state 1) = (d1, a1) depicts (d1, a1), which is a combination of the detection and attack strategies when the sink node transits from state *st* to *st+1*, and the reward established by *x* is defined as R1(fx(st)), which is given in (Eq. 4.1):

$$R1\left(f_x(S_t)\right) = \begin{cases} 0 & \left(if\ P_d = low\ and\ Cost_{processing} = low\right)\ or\ \left(if\ IR = low\ and\ Cost_{processing} = low\right) \\ R1 & \left(if\ P_d = high\ and\ Cost_{processing} = low\right)\ or\ \left(if\ IR = high\ and\ Cost_{processing} = low\right) \\ -R1 & \left(if\ P_d = low\ and\ Cost_{processing} = high\right) or\ \left(if\ IR = low\ and\ Cost_{processing} = low\right) \end{cases} \quad (4.1)$$

In the first case of Eq.4, no detection and no attack are defined. Accordingly, the reward is fixed at 0. The second case is when the sink node detects an attack with high accuracy, and its reward is R1. In the last case,

$$\left(if\ P_d = low\ and\ Cost_{processing} = high\right)\ or\ \left(if\ IR = low\ and\ Cost_{processing} = low\right)$$

, where the sink node employs strategy P_d with low processing cost and high detection accuracy to identify attack strategy IR with low attack and low processing cost, the reward is –R1. The first term, (P_d), represents the gain of employing the sink node's strategy to detect attack strategy *ai*,

and ($Cost_{processing}$) represents the cost of using the strategy. The second term, (IR), represents the gain of utilizing the attack strategy with the processing cost for the attacker.

In the IDS2 scenario, the reward function incorporates the shield policy and attack strategy when the BS transits from state *st* to *st+1*, and the reward received by the base station is defined as R2(fx(st)), as given in (Eq. 4.2):

$$R2(f_x(S_t)) = \begin{cases} 0 & (if\ P_k = low\ and\ Cost_{defend} = low)\ or\ (if\ IR = low\ and\ Cost_{processing} = low) \\ R2 & (if\ P_k = high\ and\ Cost_{defend} = low) or\ (if\ IR = high\ and\ Cost_{processing} = low) \\ -R2 & (if\ P_k = low\ and\ Cost_{defend} = high) or\ (if\ IR = high\ and\ Cost_{processing} = low) \end{cases} \quad (4.2)$$

In Eq. 4.2, the first case signifies no defence and no attack. Therefore, the reward is set to 0. In case two, when the base station defends against an attack with high defence strategy, its reward is R2. The last case indicates that the base station uses strategy P_k with high processing cost and low cost of defending against an attack strategy, therefore the reward is ‑R2. The first term (P_k) represents the base station's gain of using the strategy to eradicate attack strategy *ai*, and ($Cost_{processing}$) signifies the cost of using the strategy. The second term (IR) denotes the gain of applying the attack strategy with the processing cost for the attacker.

It is assumed that the state of node *x* is *s0* at t = 0. If the defence and detection strategies *d1, d2* are taken against an attack strategy *a*, the state of node x evolves from *s0* to *s1*, and node *x* (with respect to the sink node and base station) receives a reward R (fx(s0)) and so on (Eq.6). In Q-learning, the state of node *x* transits from *s*0 to *s*1 and eventually to *sp* where $1 \ll p \ll k-1$, where *k* signifies the efficiency of IDS1 using the *di* strategy in detecting and defending against an *aj* attack strategy. Thus, the accumulated reward received by *x* is:

$$R_x^p = \sum_{t=0}^{p} \gamma^p \; R\left(f_x(s_p)\right) \tag{4.3}$$

where $\gamma \in [0, 1)$ is the discount rate parameter. An attack strategy, and the objective of IDS2, is to select a suitable defence policy against an assault to accumulate rewards. It is worth noting that R_x^p will be calculated as two sub-rewards, such as R1 for the base station and R2 for the sink node. An instance of the reward function given to the cluster head (sink node) and attacker is the total amount of positive reward signals received when no attack has occurred and no alarm is raised (True Negative), and the number of correct invasion cases detected by the system (True Positive). The game theory phases include:

- **Phase 1:** The sink node monitors message attacks through the game-based FQL operation as the first step defined by IDS1 (see Table 4.2), after which it conveys the message to the base station for the second step function defined by IDS2 (Table 4.3).
- **Phase 2:** Upon receiving an abnormal signal from the sink node, IDS2 applies its detection fitness test in conjunction with the knowledge database to assess attack patterns and severity. This evaluation permits IDS2 to regulate the overall defence strategy in order to mitigate the DDoS attack. The IDS2 function uses the fuzzy game theory principle to select an appropriate defence tactic to shield the message-consuming sensor node. The IDS2 also informs the affected sink node that it needs to protect itself against the offending attack pattern.
- **Phase 3:** The sink node verifies the current state of IDS play with the sensor node. If the sink node still detects an irregularity, it is likely that the IDS2 operation opted for the wrong defence strategy, and consequently, the sink node advises the IDS2 to revise its detection strategy. If the attack pattern alert count at the sensor node decreases in number, the sink node systematically endeavours

to confirm the current state of IDS play with the sensor node until the attack condition is resolved and returns to the correct defence strategy state.

- **Phase 4:** The sink node notifies IDS2 that the attack at the sensor node has been successfully counteracted and the attack has ceased.
- **Phase 5:** The IDS2 thus concludes defending the sensor node.

Utility function

To appraise the efficacy of the links determined by G-FQL and to determine the rule applicability at every point in time, Eq.(4.4) was utilized in this thesis, as suggested by Liao et al. (Huang et al., 2013). In Table 4.6 the utility function parameters are described.

$$U = \rho * SP - \beta * FN - \theta * FP \quad (4.4)$$

Table 4.6: Utility function parameters

Parameters	Explanation
U	Is a utility
ρ	Symbolizes the weight of effective prediction, $q = 0.75$
SP	Characterizes the true confidence rate of attack patterns.
β	Signifies the weight of failed estimates (attack but no defense), b = 1
FN	Represents false negative of attack patterns - there are attacks but no defense
θ	Denotes the weight of failed predictions (defense but no attack), $h = 1$
FP	Represents false positive of attack patterns - there is defense but no attack

The game principle approach entails detection accuracy with low time complexity, which only subsequently begins to formulate a shield policy. The major weakness of the game theory is that if attacks recur over a short period, a considerable amount of time is consumed in the detection phase, something that deteriorates the defence. It can be said that the detection precision is low while the false alert rate is high. This problem is a worst-case scenario, but it can be addressed using the

Cooperative-FQL. Its principal contribution is identifying the probability of future attacks aimed at a wireless sensor node. For frequent attacks occurring over a short time, multi agent-based FQL was adopted to handle the excessive time spent on detection. The aim of the proposed FQL is to obtain high detection accuracy with a low false alarm rate.

4.2.2 Fuzzy Q-learning algorithm

To overcome the required complex detection and defence time as well as detection precision issues in our game theory method, the FQL algorithm is applied in this thesis to detect probable future points of attack beforehand. To optimize Q-learning algorithm performance from the action selection method and reward function perspectives, fuzzy min-max methods are employed. In the proposed scheme, the fuzzy min-max action selection and reward function with conventional Q-learning are evaluated. High detection accuracy performance was revealed. For this reason, FQL is employed to reinforce a system's learning capability.

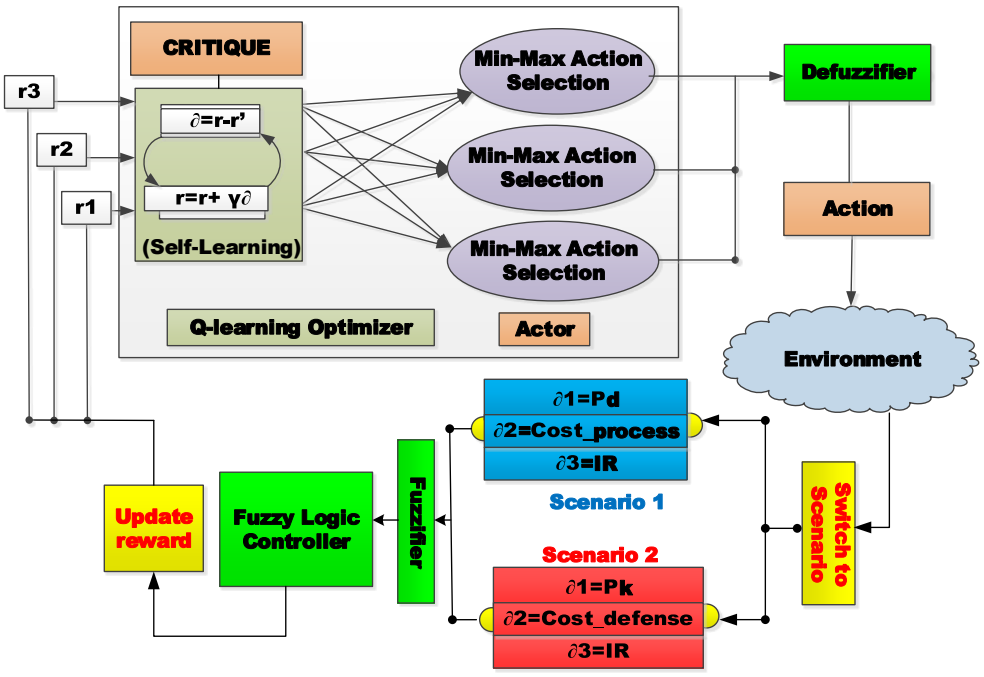

Figure 4.4: Block diagram of the FQL optimization system

The FLC inputs are provided in two scenarios through the switching process. In the first scenario, which is a game between a sink node and attacker, we have P_d=$\left(\frac{Correct\ \ attack\ detection}{Total\ detection\ and\ no\ detection}\right)$, the cost of attack detection during sink processing ($Cost_{process\ detect}$) as per the sink node utility function and $IR = \left(\frac{Number\ of\ malicious\ attacks}{Total\ malicious\ attacks\ sent}\right)$ as well as $Cost_{processing} = processing\ time\ for\ attack$ with respect to the attacker utility function. These correspond to the fuzzy state of network S1 (t) from the first scenario $S_1(t) = [Pd, Cost\_process, IR]$. In the second scenario, the game between the base station and an attacker, $Pk = cost\ of\ killing\ attack$, Cost defend = $Power\ cost\ during\ defence\ against\ attack$ adapts as a base station utility function while $IR = \left(\frac{Number\ of\ malicious\ attacks}{Total\ malicious\ attacks\ sent}\right)$ and $Cost_{processing} = processing\ time\ for\ attack$, regarding the attacker utility function, correspond to the fuzzy state of network S2(t) from the first scenario: $S_2(t) = [Pk, Cost\_defend, IR]$.

The FLC output is given by the increment in states and represents the action of the sink node and the base station A(t). The reward signal, R (t), is built from FLC and is measured in both modes of adjacency to test if the sensors experience attacks in detection mode and the base station correctly defends against attacks. The linguistic variables Pd, Cost_ Process, and IR act as inputs for the first scenario, while the linguistic variables Pk, Cost_defend, and IR serve as inputs for the second scenario.

The Detect Confidence (DC1) behaves as output for the first scenario and the Defend Confidence (DC2) acts as output for the second scenario. They are both applied in the experiments (Table 4.7).

Table 4.7: Linguistic variables for fuzzy set input and output

Type of Scenario	**Variable**	**Attribute**	**Membership function**		
Attacker and sink node	Input	*Pd*	Low	Med	High
		Cost process	Low	Med	High
		IR	Low	Med	High
	Output	*Detection Confidence (DC1)*	Low	Med	High
Attacker and base station	Input	*Pk*	Low	Med	High
		Cost defence	Low	Med	High
		IR	Low	Med	High
	Output	*Defence Confidence (DC2)*	Low	Med	High

Two fuzzy sets are identified in all inputs and outputs, whose linguistic terms are 'Low' (L) and 'High' (H). The fuzzy reward was elaborated in Section 4.1.3. Hence, the objective is to determine the total reward value over time. If the defence and detect strategy *di* is used against attack strategy *aj* at time *p* and the state of node *x* transits from *St* to *Sp+1*, the Q-learning function for IDS1 is Q: S×D×A→R as in:

$$Q(S_p, d_i, a_j) \leftarrow Q(S_p, d_i, a_j) + \alpha\left[R\left(f_x(s_p)\right) + \gamma R_x^{p+1} - Q(S_p, d_i, a_j)\right] \quad (4.5)$$

where $\alpha \in (0, 1]$ is the learning rate factor. In this scheme, the Q-function is applied in dual situations, such as IDS1 and IDS2. In each state, the reward function rewards the cluster head (sink node) using the Q-learning method and the base station also obtains the reward. G-FQL attains the final reward value of each player. A learning rate of zero means the system no longer learns anything new, but a value of 1 would prompt the system to adjust its accuracy strategy as it self-learns from new attacks and to update the information in its knowledge base. If the reward value is below the threshold v, FQL IDS1 deems node x secure; otherwise, it considers the node insecure and takes suitable detection action against the attack. Coinciding with this evaluation, FQL IDS2 takes appropriate defensive action against any potential attacks.

4.3 Chapter Summary

In this chapter, the interaction between attackers, sink nodes and the base station was studied, after which a novel, Game-based FQL, cooperative game theoretic defence mechanism was proposed. This system combines the cooperative-based game theory with fuzzy Q-learning algorithmic elements. As such, the collaboration between the detection sink node player and response base station players is reinforced to defend against an incoming DDoS attack that may cause congestion and downtime in network communication due to flooding packets. The Game-FQL model is a triple-player game strategy construed as two-player, providing double defence against a single attacker. It adds confidence and establishes a reputation as extremely apt in tracking attackers and defending the system. This strategy-based cooperative game adapts to continuous self-learning from past attacks and the fuzzy Q-learning decision making process behaviour to defeat attackers. By defining incentives for cooperation and disincentives for fraudulent behaviour, it has been determined that repeated interaction sustains cooperation, builds confidence and enhances reputation, another benefit of Game-FQL. Game theory-based Fuzzy Q-learning (Game-FQL), a mechanism in IDPS, is an invaluable tool for progressively securing next-generation complex heterogeneous computing and networking environments against sophisticated attacks and attackers, beyond what is encountered today. A future initiative could be to extend the proposed Game-FQL mechanism by incorporating data from various attack types and sources to further enhance its decision making capabilities in order to impede existing or new attacks.

Chapter 5 : FRAMEWORK EVALUATION

This chapter reports on the data collection method for the evaluation of proposed Game based IDPS framework. It explains the tools used for testing the proposed framework, data generating technique and the statistical method used for the processing of data.

The chapter is organized into nine sections. Section 5.2 explains the experimental setup and programming tools used for the implementation and testing of the proposed Game based FQL- IDPS framework and the statistical method used for the compilation of empirical data. Section 5.3 presents the data generation and analysing the flood attack strategy in evaluating the execution of IIDPS on network. Section 5.4 summarizes data analysis of the game based FQL in evaluating IIDPS in terms of detection accuracy.

Section 5.5 presents the analysis of game based FQL for testing the defense rate of G-FQL framework. Section 5.6 presents the analysis for number of live nodes during detection and prevention. Section 5.7 evaluates the energy consumption over time and Section 5.8 analyses the energy consumed by different deployed nodes in G-FQL IDPS. Finally, Section 5.9 extracts conclusive remarks.

5.1 Simulation and analysis

5.1.1 General tools

To carry out the experiments in the different evaluation stages, this study used open source simulation software, namely network simulator version 2 (NS2). The reason for utilising the applications was their openness and public availability, as well as being free to use. The descriptions of the applications are briefly explained as follows:

In 1996-97, network simulator version 2 (NS2) was initiated based on a refactoring by Steve McCanne(Group, 2004). Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented language of Tcl. The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language. Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution (many forks of ns-2 exist, both maintained and unmaintained). It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X and Windows versions that support Cygwin. It is licensed for use under version 2 of the GNU General Public License.

5.1.2 Design Assumptions

To facilitate and conduct experiments in this thesis, along with the specific security model in the proposed framework, specifically, given the lack of specific information on resources, some assumptions had to be made to the attack and game players' scenario.

The first assumption is given for players in our scenario. In this scheme, we assigned the player one as a base station, player two as a sink node, and player three as

an attacker (See Chapter 4). The second assumption is to create a routing protocol. In this thesis, a LEACH protocol was adopted and the agents or nodes communicate through this protocol. Distributed Denial-of-Service (DDoS) attack is considered as a third assumption. Due to lack of real dataset, in this thesis, a generator function is defined to create flooding attacks during a period of time.

5.1.3 Simulation Setup

The Low Energy Adaptive Clustering Hierarchy (LEACH) protocol was utilized in the simulation, as it most closely reflects WSN in practice and it is also capable of dealing with energy consumption concerns in WSNs. The simulations were run for 1000s with LEACH as the routing protocol, the initial access point energy was 100 joules, the effective transmission range of the wireless radio for the access point was 100m, the sink node transmission range was 100m, the common node transmission range was 50m and the transport protocol is given in Figure 5.1. In addition, the cooperative game-based IDPS with fuzzy Q-learning was employed to hasten the simulation.

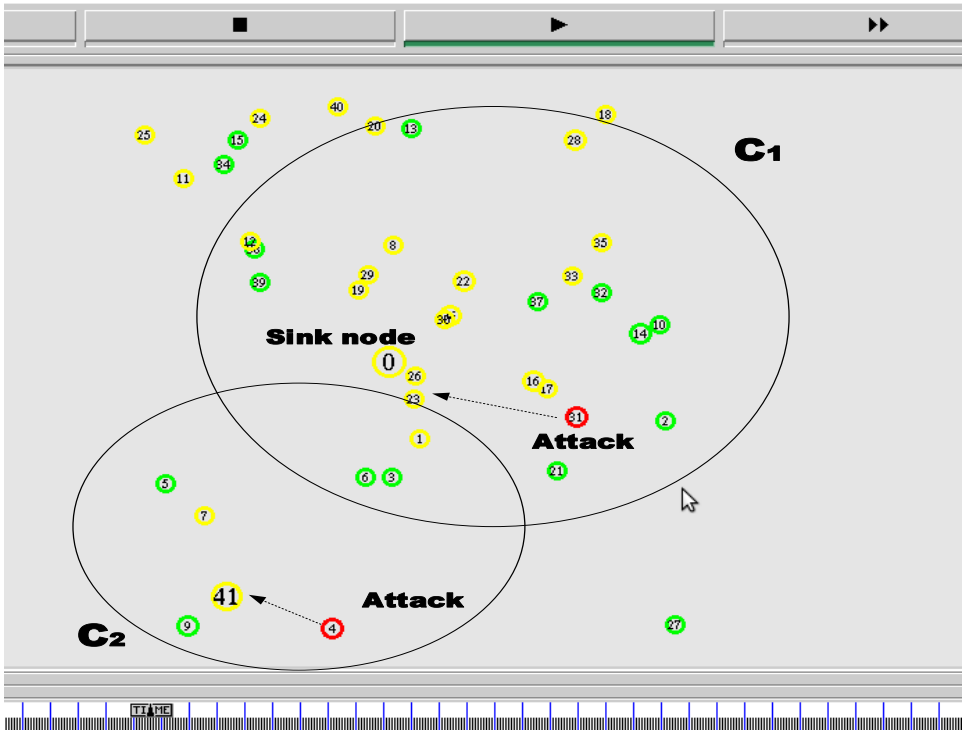

Figure 5.1: Simulated WSN environment

Table 5.1 presents the WSN configuration along with the set of parameters applied in NS-2. However, in practical WSN security operation, minimizing energy usage to conserve energy and maximize detection accuracy as much as possible is vital when designing and running G-FQL IDPS. The results obtained from the proposed algorithm are compared with those from Fuzzy Logic Controller, Q-learning, and Fuzzy Q-learning as well as the Markovian Game (Huang et al., 2013) .

Table 5.1: Wireless sensor network parameters in NS-2

Wireless Sensor Network Parameters	**Values**
Access Point	1
Common Nodes	200
Sink Node in each Cluster	1
Routing Protocol	LEACH
Scenario Size	100*100
Simulation Time	1000s
Transport Protocol	UDP
Access Point Initial Energy	100 joules
Access Point Transmission Range	100 meters
Sink Node Initial Energy	10 joules
Sink Node Transmission Range	70 meters
Common Node Initial Energy	10 joules
Common Node Transmission Range	50 meters

5.1.4 Generating and analysing the flood attack strategy

The purpose of this section is to analyse the quantitative behaviour of attacks in the UDP protocol layer. In the present experiments, normal UDP traffic was initially considered, after which the attack intensity under flood attacks with UDP traffic was explored. Subsequently, the total energy consumed before and after attack was examined. The accuracy of detection and defense as a result of executing the G-FQL algorithm was finally assessed. To generate an attack, a random function was employed, which selected subject nodes from each cluster to attack. The selected nodes adjusted their functions to send flooding packets to the cluster head. Algorithm 5.1 displays the attack strategy.

Algorithm 5.1: Attack strategy

1. Start
2. Min(r)=0 %% Initial round simulation (Max(r)=n)
3. While (r<>n)
4. Decide r round's cluster head randomly
5. Cluster head advertises schedule time to all its common nodes
6. Generate attack node randomly
7. Attack node receives schedule time message from its cluster head
8. Attack node starts to compromise victims
8.1. Attack node sends flooding packets to its cluster head in this round
8.2. Victim (cluster head) receives data more quickly than normal state, so its energy will decrease rapidly
9. End.

In the experiment, an attack with UDP attack intensity was implemented. Figure 5.2 indicates flooding attack intensity per packet length. Greater attack intensity percentage obviously occurred between 200 and 300s, at which time packet length also reached elevated values. In Figure 5.3 it appears that UDP attack intensity affected the WSN energy, besides the fact that energy was consumed in proportion to attack intensity. For example, for attack intensity between 100 and 150s, the most energy was consumed.

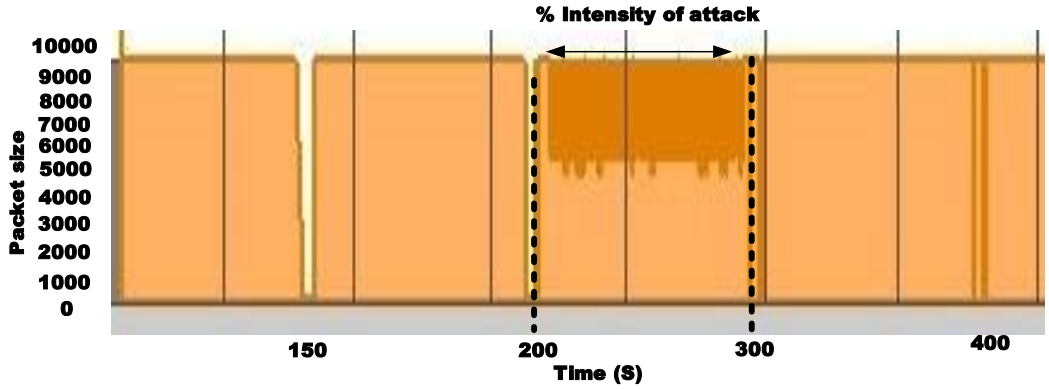

Figure 5.2: Effects of UDP attack intensity on packet size

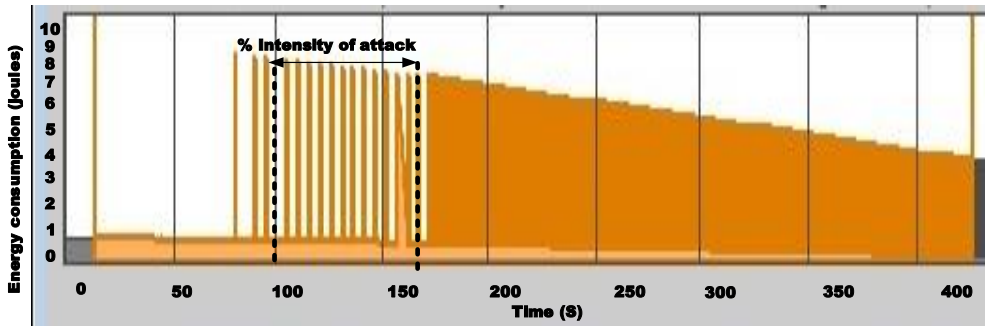

Figure 5.3: Victim node's energy level over time

In the present research work, three sets of experiments were conducted to examine the effects of attack detection accuracy and defense rate against attacks based on the Fuzzy Logic Controller, Q-learning algorithm, Fuzzy Q-learning and Game theory-based Fuzzy Q-learning algorithms. The cost function was calculated according to Eq. 4.7.

5.1.5 Analysis of the game-based FQL IDPS in terms of detection accuracy

The proposed game-based Fuzzy Q-learning (G-FQL) algorithm with the cost function $U = \rho * SP - \beta * FN - \theta * FP$ was compared with existing soft computing methods (Fuzzy Logic Controller, Q-learning, and Fuzzy Q-learning) with respect to the attack detection precision of modeled Denial-of-Service attacks. A comparison between the average utility function and G-FQL with cost maximization indicates that the latter yielded an improvement of 3.29% with 1.86 standard deviation as opposed to the FQL algorithm with 0.83 (Table 5.2).

Table 5.2: Simulation results of the detection algorithm for DDoS attacks

Percentage of Attack (%)	FLC				Q-learning				FQL				Game-based FQL			
	SP %	FP %	FN %	Utility Function	SP %	FP %	FN %	Utility Function	SP %	FP %	FN %	Utility Function	SP %	FP %	FN %	Utility Function
1	49.50	1.90	2.40	56.38	76.00	1.40	1.20	54.40	80.10	1.20	1.10	57.78	83.20	1.20	1.10	60.10
5	49.80	1.98	2.80	56.07	76.70	1.60	1.40	54.53	81.20	1.40	1.30	58.20	83.40	1.30	1.20	60.05
10	50.01	2.00	3.20	55.71	76.90	1.90	1.70	54.08	82.50	1.90	1.70	58.28	84.30	1.50	1.60	60.13
15	51.20	2.04	3.60	56.56	77.80	2.10	2.00	54.25	83.70	2.10	2.00	58.68	85.60	1.70	1.80	60.70
20	50.90	2.40	3.90	55.38	78.00	2.40	2.20	53.90	83.90	2.40	2.20	58.33	87.90	1.90	2.00	62.03
25	51.90	2.80	4.10	55.93	78.90	3.10	2.70	53.38	84.20	2.60	2.30	58.25	88.30	2.10	2.30	61.83
30	52.70	2.90	4.20	56.68	80.20	3.40	3.00	53.75	85.80	2.80	2.60	58.95	89.70	2.40	2.50	62.38
35	49.40	3.00	4.70	51.70	82.80	3.90	3.20	55.00	86.40	2.90	2.70	59.30	90.50	2.60	2.70	62.58
40	49.50	3.01	5.00	51.37	82.90	4.20	3.80	54.18	87.70	3.20	3.00	59.58	91.70	3.10	3.00	62.68
45	50.02	3.20	5.30	51.38	83.70	4.90	4.10	53.78	88.50	3.40	3.20	59.78	92.40	3.20	3.40	62.70
50	51.04	3.50	5.60	51.90	83.90	5.20	4.80	52.93	89.60	3.90	3.50	59.80	94.20	3.30	3.70	63.65
55	50.30	3.70	5.80	50.48	84.90	5.60	5.10	52.98	90.40	4.10	4.00	59.70	96.50	3.50	3.80	65.08
60	49.30	3.70	5.90	49.08	85.00	5.80	5.70	52.25	92.40	4.50	4.30	60.50	98.20	3.70	3.90	66.05
Average	51.20	2.78	4.35	53.74	80.59	3.50	3.15	53.80	85.88	2.80	2.60	59.01	89.68	2.42	2.54	62.30
Std. Dev.	1.03	0.66	1.15	2.76	3.37	1.50	1.47	0.75	3.71	1.01	0.98	0.83	4.86	0.87	0.98	1.86

It is evident that G-FQL with a cooperative mechanism attained the utmost detection accuracy gain. It can also be inferred from Figure 5.4 that detection accuracy per percentage of attack is higher with the G-FQL algorithm than the other methods.

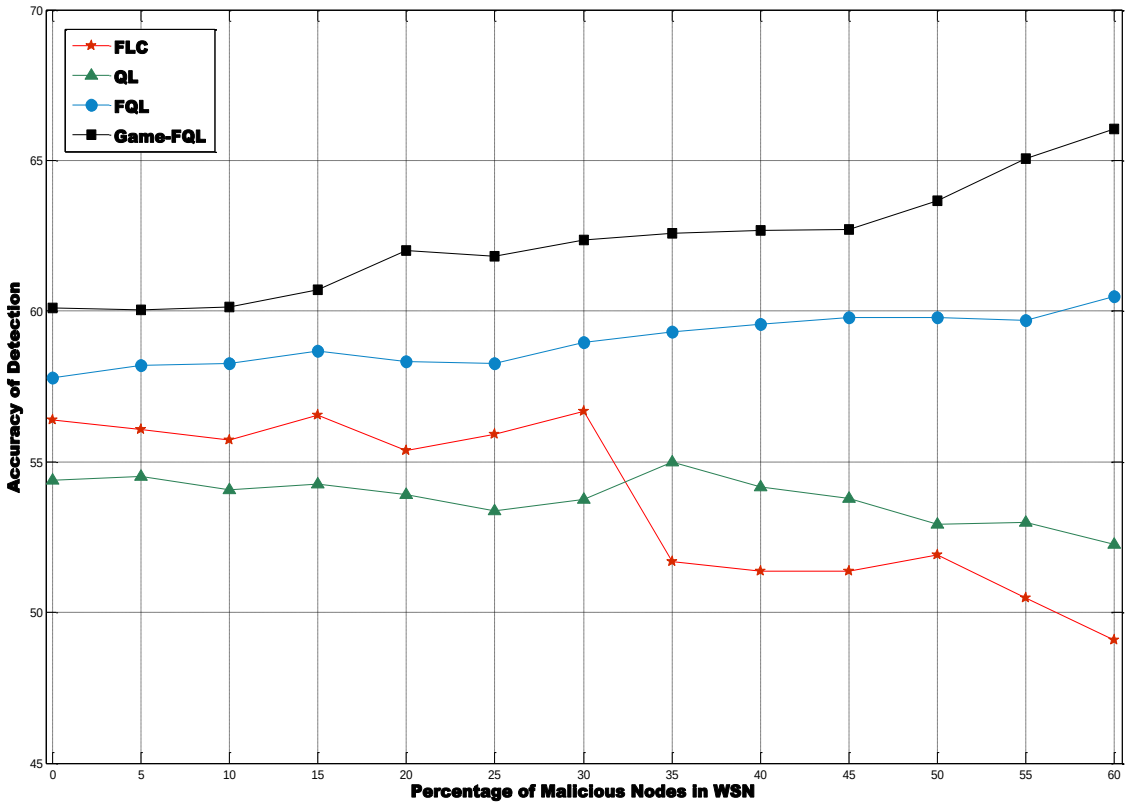

Figure 5.4: Comparison of detection accuracy values

In Figure 5.4, the X-axis shows the percentage of malicious nodes in an attack, and the Y-axis indicates the accuracy rate. At higher attack frequencies, the proposed method (Game-based FQL) displays greater accuracy scores.

5.1.6 Analysis of game-based FQL IDPS in terms of Defense Rate

The proposed Game-FQL method was weighed against that of Huang et al. (Huang et al., 2013), who used the game theory and Markovian IDS with an attack-pattern-mining algorithm. According to Huang et al.'s (Huang et al., 2013) empirical results, the defense rate effectiveness of non-cooperative-based Markovian IDS with an attack-pattern mining algorithm for 60% of malicious nodes in a network and two sink nodes ranged between 72% and 97% (Figure 4.9). With the proposed game-based FQL IDPS, the successful defense rate was between 79% and 98%, as per Figure 5.5 as well. It can be concluded that integrating the game theory with the Fuzzy Q-learning algorithm outperforms individual defense schemes.

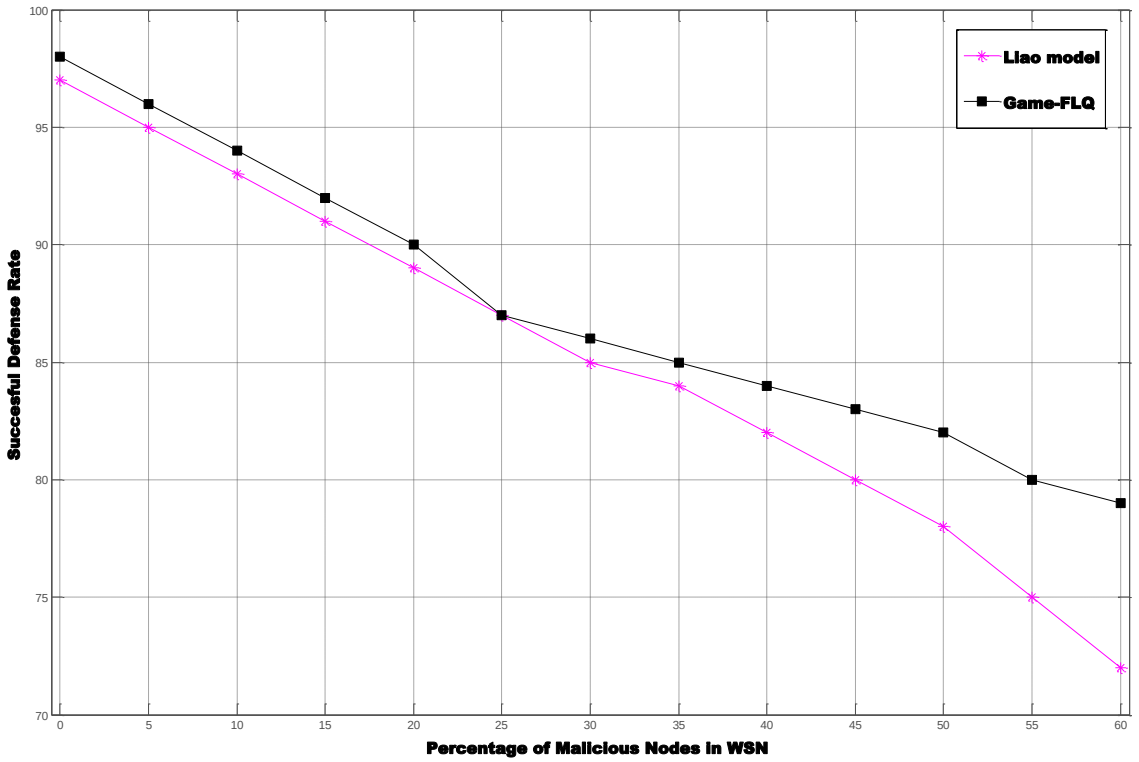

Figure 5.5: Game-based FQL in terms of accuracy of defense rate under attack trends

Figure 5.5 points out that the successful defense rate values for Huang et al.'s model (Huang et al., 2013) and the proposed methods decreased from 100% to 87% when the anomaly percentage increased. However, the proposed method gained the advantage of a successful defense rate due to the higher percentage of malicious nodes detected compared to Huang's lower success rate. It can thus be deduced that by integrating the game theory with the Fuzzy Q-training method, performance surpasses that of any other individual defense approach.

5.1.7 Analysis of game-based FQL IDPS in terms of number of live nodes

This experiment was conducted to evaluate the performance of the Game-FQL algorithm in terms of number of live nodes during the simulation runtime. In the current scheme, the number of sensor nodes was 200. Figure 5.6 displays the number of live nodes for different algorithms throughout simulation runtime. The simulation outcomes indicate the number of live nodes at the end of the simulation time (1000s), according to which, the number of live sensor nodes in the proposed Game-FQL method is

significantly greater than existing algorithms. Game-FQL maintains 50 live nodes against an attack in comparison to 42, 32, and 21 live nodes for FQL, QL, and FLC, respectively.

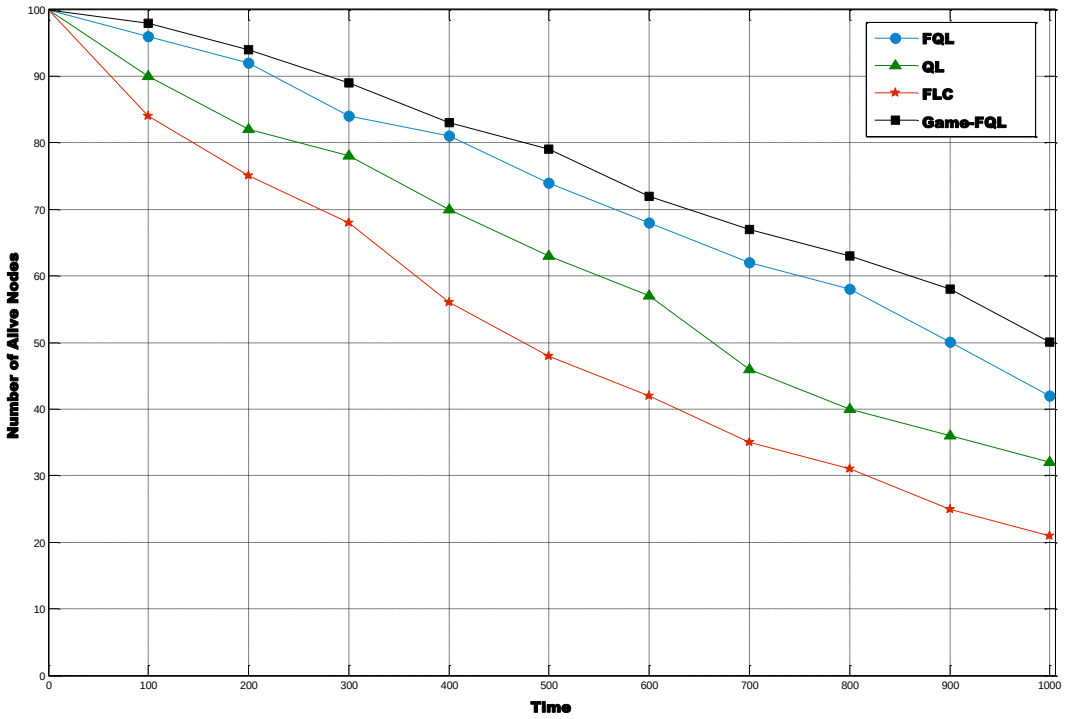

Figure 5.6: Number of live sensor nodes during simulation runtime (ms)

The procedure of adjusting rules according to FLC-based DDoS attacks is more time-consuming, and the attacker defeats a high number of nodes during FLC detection (Baig et al., 2010). Q-learning-based DDoS attack detection is capable of handling minor-class DDoS attacks, but the multi objective procedure or major features of a DDoS attack consume maximum resources, especially in a real-time environment (Liu, 2008). Fuzzy Q-learning-based DDoS attack detection utilized the min-max fuzzy method to enhance the classification scheme. The min-max fuzzy classifiers perform well with a reduced dataset, but inaccurately when the high volume of traffic increases further and the fuzzy IDPS may crash. In addition, prior knowledge of data distribution is required for the FQL algorithm. In the Fuzzy Q-learning algorithm, observation is limited by one single classifier. Therefore, this algorithm fails due to high volumes of real-time traffic. In the currently proposed method, the cooperative policy evaluates the

proficiency of an agent to optimize the cost function based on weight assignment mechanisms for real-time DDoS attack detection. The countermeasure mechanisms result as modules to be applied in Game-FQL architecture and system implementation to accelerate the detection and defense learning process in a fraction of the usual time. Thus, the Game-FQL preserves a greater number of sensor nodes during simulation.

5.1.8 Analysis of game-based FQL IDPS in terms of energy consumption over time

In this experiment, the energy consumed by the Game-FQL algorithm during DDoS attacks on sensor nodes in comparison to FLC, QL, and FQL is studied. Figure 5.7 provides the comparison between the mentioned algorithms in terms of total energy consumed by sensor nodes.

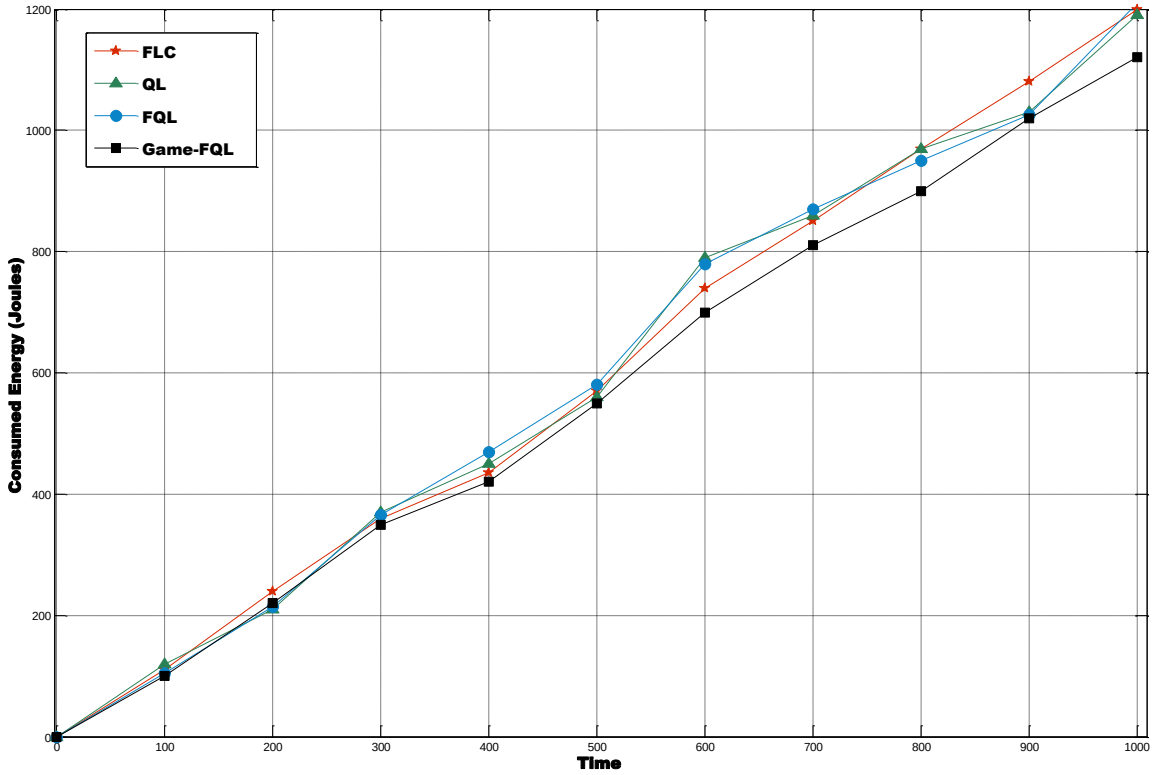

Figure 5.7: Total energy consumption versus number of sensor nodes under malicious attack

In existing detection, the players (sink node and base station) partake in activities such as local sensing and data reporting, which consume additional energy. The overhead of energy consumption may be considerable if the number of cooperating players or the amount of sensing results in the report is very large. Thus, energy

efficiency needs to be considered in cooperative sensing schemes. To address this issue, the cooperative game-based FQL method enhances energy efficiency via optimization.

5.1.9 Analysis of the energy consumed by different deployed nodes in the game-FQL

The impact of number of deployed sensor nodes on energy consumption is shown in Figure 5.8. It is observed that with an increasing percentage of deployed nodes, the proposed Game-FQL is able to consume the total amount of energy in comparison with FQL, QL, and FLC.

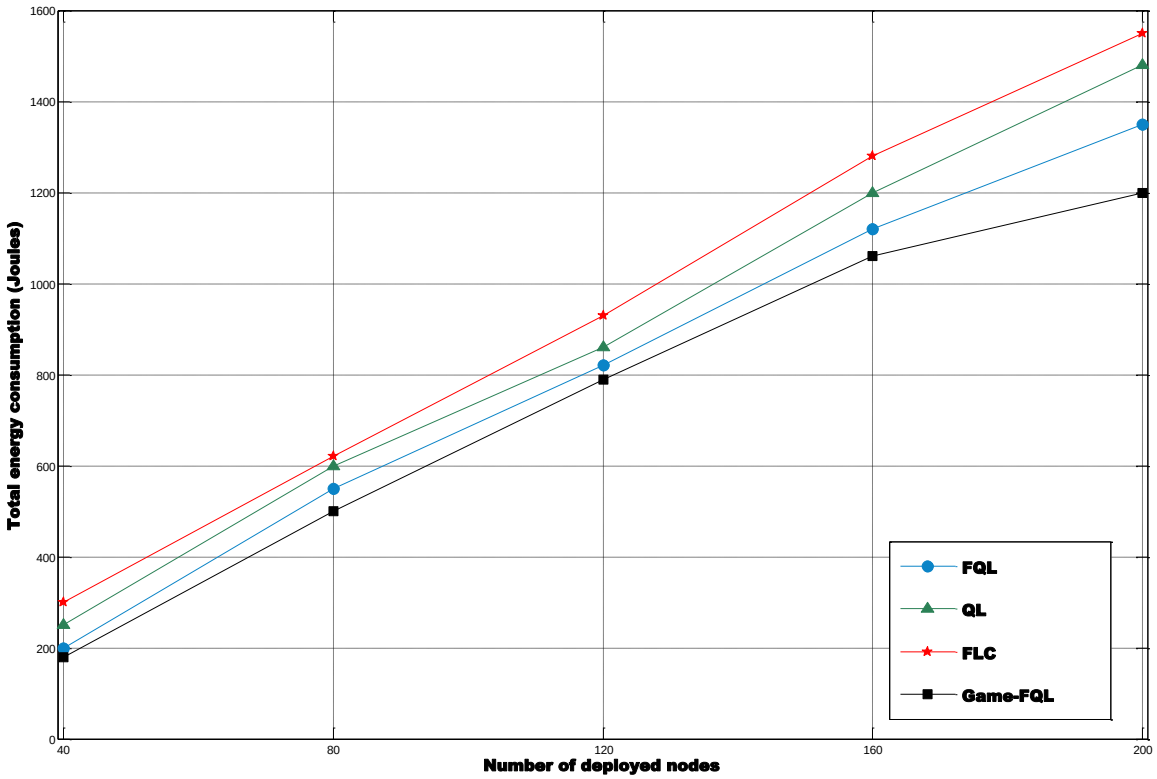

Figure 5.8:Total energy consumption versus number of sensors deployed in a network

Finally, Figure 5.8 depicts the total energy consumed with varying numbers of sensor nodes deployed in the network. The experiment was run for 40, 80, 120, 160, and 200 nodes. As expected, when more nodes are present in the network, the energy consumption rate is lower than other comparable methods. This is attributed to the fact that the proposed Game-based FQL agents prefer to maximize their own utility function by means of cooperating learning algorithm to avoid the energy consumption by sensors from each cluster. However, it would be interesting for the cooperative Game-FQL

solution to be implemented, for instance, to “BEE-C: A bio-inspired energy efficient cluster-based algorithm for data continuous dissemination in Wireless Sensor Networks” (da Silva Rego et al., 2012), to verify the energy consumption for intrusion detection and prevention.

5.1.10 Analysis of the computational time in the game-FQL

Preprocessing time includes the time spent in feature extraction and normalization. The training time depends on the number of times the classifier needs training which in turn depends on the mean square error between iterations reaching goal minimum. Testing time includes the time spent in testing the unlabeled instances by weighted mean. Table 5.3 shows the performance comparison of the G-FQL in terms of consuming time obtained during the experiments. From Table 5.3, it can be realized that the training time of G-FQL is similar to FQL, but it consumes more testing time than the FLC, Q-learning, and FQL. Also, the computational time was calculated on Intel 3.10 GHz, Core i-5 Processor, 4 GB RAM computer.

Table 5.3 : Performance comparison of G-FQL in terms of consuming time

Dataset	**Algorithms**	**Training time (seconds)**	**Testing time (seconds)**
Real data	Fuzzy Logic Controller	3.10	1.30
	Q-learning	3.14	1.36
	Fuzzy Q-learning	3.22	1.40
	G -FQL	**3.22**	**1.42**

Testing time of the proposed G-FQL method is a little high due to the ensemble output combination methods such as fuzzy Q-learning and weight strategy sharing algorithm, but more detection accuracy was achieved in G-FQL. The speedup of G-FQL can be improved when a hybrid classifier is executed in parallel processors. Thus, all the modules can be processed in parallel by different engines in order to reduce the overall processing time considerably.

5.2 Chapter Summary

The first stage of the evaluation study has shown the statistical analysis of flooding attack with generate attack algorithm. The attack model was introduced as a means to estimate the damage of flooding attack. Likelihood of attack intensity per packet length and attack intensity affected the energy. With a combination of the fuzzy reinforcement algorithm and the aid of game theory, the detection for attack is improved.

The second stage aims to investigate the effectiveness of proposed model as a strategy in detection. One of the criteria to support the detection system is to consider the ability of agents in order to share their knowledge to identify attacks. For instance, intelligent detector identifies a DDoS attack with a fast ability of detection in order to minimise its impact. This intelligent and fast detection process is crucial to the proposed framework, as a good detection strategy increase the ability of model in facilitation a cooperative based intelligent mode. Therefore, in order to satisfy such claims, this stage investigates the ability of proposed frameworks to response strategy model.

The third stage also investigates the relationship between attacked and their classification (e.g. accuracy of detection/ false alarm rate). With the implementation of collaborative security strategy in attack detection and response, some improvements have been made. In practical, a better cost functions for attack detection can be produced compared to the cost function used in the traditional methods. In addition, there is a reduction in terms of false alarm rate of attacks that need to be marked by fuzzy labelling. This fuzzy labelling allows security experts to classify number of attacks and type of damage of attacks in order to response only to an appreciate attacks; hence it could save time and responses.

Chapter 6 : CONCLUSION

This chapter summarizes the major finding of the study by reviewing the achievements of the research. In particular, the first section of this chapter highlights its most important findings, as well as its limitations. The next section dedicates the main contributions of this thesis and how the objectives have been achieved. Finally, this chapter concludes with a description of possible future work in the topics covered and showing how the proposed framework could be enhanced in the future.

6.1 Achievements of the study

This thesis launched with an investigation into the different types of intelligent intrusion detection and response systems, exploring issues related to the IIDPSs and Co-IIDPS in networks. The study proposed a novel collaborative game based IDPS framework in order to identify the distributed denial of service attacks and to show the ability of multi agent based computational intelligence methods in terms of collaborative IIDPS. The proposed model was compared with existing soft computing methods (fuzzy logic controller, Q-learning, and fuzzy Q-learning) with respect to the attack detection precision of modeled denial-of-service attacks. Several analyses were explored and their capabilities evaluated in order to satisfy the aims of this thesis.

The overall goal of this thesis is to establish a novel approach to identify distributed denial of service attacks and response to attackers in network environments. Within the proposed framework, which included experiments, this thesis has been successful. Details are as follows:

- ***A collaborative model for Intelligent Intrusion Detection and Prevention Systems.***

This thesis introduced three classes of IIDPS detection methodologies, approaches and technologies. Each technique has its advantages and limitations. The

TAI-based IDPS is straightforward to implement and very effective in inspecting known attacks. Still, the approach hardly identifies unknown attacks, attacks concealed by evasion techniques and several variants of known attacks. A number of fuzzy rule-based approaches to detect unknown attacks were also proposed. Such techniques may unnecessarily result in issues with excessive computing time consumption and rapid updating of the knowledge base, hindering attack effectiveness.

A more accurate and simplified approach is still required to increase efficiency and effectiveness further. Computational intelligence-based approaches such as Reinforcement Learning (RL) algorithm have the merit of possessing with no prior knowledge of attacks. They do not work well in real-time applications due to the high computational complexity. A multi agent-based CI (MCI) not only mitigates high computational complexity such as time consumption and updating knowledge, but also enhances detection performance (See Chapter 2).

Thus, the collaborative management using the multi agent system-based computational intelligence portrays the ability to mitigate detection problems. In other words, the individual or single capabilities in terms of self-cooperative techniques (without using CI methods) consider all the features addressed in their systems. These inefficiencies are evidence of the lack of cooperative knowledge regarding suitable CI methods to identify intrusion prior to initiating any development.

All new solutions to developing multi agent-based CI methods consider the requirements (detection and false alarm rate) as being able to overcome Cooperative-IIDPS complexities and meet the real operational goals of networks.

A novel collaborative-based IIDPS (Co-IIDPS) architecture is proposed and presented. It demonstrates the impact of a Multi Agent System-based computational intelligence (MCI) technique on enhancing the efficiency of detection and false alarm

rates. In other words, incorporating a multi- agent system (MAS) to computational intelligence (MCI) in terms of Co-IIDPS allows monitoring intrusion activity. Fuzzy system (FS) with reinforcement learning (RL) in terms of fuzzy reinforcement learning manager (FRLM) has merged into Co-IIDPS, resulting in high true positive and low false alarm rates. The policy aspect of MAS-based FRLM applies a negotiation method to improving the detection accuracy. The developed Co-IIDPS architecture around MAS-based FRLM satisfies the detection performance (See Chapter 3). This architecture portrays the clear notion of cooperative learning-based detection to satisfy the requirements of IIDPS. In conclusion, the detection management techniques can be improved by minimizing the false alarm rates and increasing the detection rates in addition to decreasing energy consumption in networks.

In the context of Co-IIDPS, adaptive game theoretic techniques are adequate for network parameter optimization due to the complexity and dynamism of networks. The main benefits of applying such techniques are cost savings and improved network performance. The model helps the proposed framework to identify different types of DDoS attacks, each of which has its own unique characteristics (see Chapter 4). The interaction between attackers, sink nodes and the base station was studied, after which a novel Game-based FQL, cooperative game theoretic defense mechanism was proposed. This system combines the cooperative-based game theory with fuzzy Q-learning algorithmic elements. As such, the cooperation between the detection sink node player and response base station players is reinforced to defend against an incoming DDoS attack that may cause congestion and downtime in network communication as a result of flooding packets.

The Game-FQL model is a triple-player game strategy construed as two-player, providing double defense against a single attacker. It adds confidence and establishes a reputation as extremely apt in tracking an attacker and defending the system. This

strategy-based cooperative game adapts to continuous self-learning of past attacks and the behaviour in the fuzzy Q-learning decision making process to overcome the attacker. By defining incentives for cooperation and disincentives for fraudulent behaviour, it has been determined that repeated interaction sustains cooperation, builds confidence and enhances reputation, something additionally offered by Game-FQL. In conclusion, Game theory-based Fuzzy Q-learning (Game-FQL), as a mechanism in IDPS, is an invaluable tool for increasingly securing next-generation complex heterogeneous computing and networking environments against sophisticated attacks and attackers, beyond what is encountered today.

- ***Issues in Collaborative IDPS studies.***

In Chapter 4, this thesis established a critical analysis of different perspectives when addressing the significant problems of the DDoS attack detection and response, as well as its challenges. With an aim to establish a IIDPS framework to DDoS attacks, several issues were exposed. By presenting the strengths and weaknesses of these issues, several intelligent IDPS and cooperative-IIDPS were identified which address the limitations of the previous approaches, by enhancing the cooperative game based learning techniques; it is more systematic in the detection and response process.

- ***Comprehensive evaluation stages for the proposed framework.***

In addressing the distributed denial of service attacks detection and response in WSNs, the proposed framework outlined several models and strategies. The objective of the evaluation is to examine the proposed framework and to decide whether it is sufficiently applicable to facilitate the detection and response action in a traffic network. The evaluation presented as follows:

The first stage of the evaluation study has shown the statistical analysis of flooding attack with generate attack algorithm. The attack model was introduced as a

means to estimate the damage of flooding attack. Likelihood of attack intensity per packet length and attack intensity affected the energy. With a combination of the fuzzy reinforcement algorithm and the aid of game theory, the detection for attack is improved.

The second stage aims to investigate the effectiveness of proposed model as a strategy in detection. One of the criteria to support the detection system is to consider the ability of agents in order to share their knowledge to identify attacks. For instance, intelligent detector identifies a DDoS attack with a fast ability of detection in order to minimise its impact. This intelligent and fast detection process is crucial to the proposed framework, as a good detection strategy increase the ability of model in facilitation a cooperative based intelligent mode. Therefore, in order to satisfy such claims, this stage investigates the ability of proposed frameworks to response strategy model.

The third stage also investigates the relationship between attacked and their classification (e.g. accuracy of detection/ false alarm rate). With the implementation of collaborative security strategy in attack detection and response, some improvements have been made. In practical, a better cost functions for attack detection can be produced compared to the cost function used in the traditional methods. In addition, there is a reduction in terms of false alarm rate of attacks that need to be marked by fuzzy labelling. This fuzzy labelling allows security experts to classify number of attacks and type of damage of attacks in order to response only to an appreciate attacks; hence it could save time and responses.

In conclusion, the proposed framework was analysed in terms of its detection accuracy and defence rate. The evaluation stage satisfied the number of live nodes, in particular the ability of the proposed game based IDPS to operate with reasonable response time and reduce false alarm response. Beside the effectiveness and

performances of proposed method, the energy consummation over time is evaluated. Finally, the percentage of energy consumed by different deployed nodes is evaluated to show the performance of proposed framework.

- ***Simulation of the proposed framework.***

To appraise the performance and check the connection between G-FQL and the routing protocol, NS-2 is simulated. In this thesis only the Distributed Denial-of-Service (DDoS) attack is considered. DDoS is characterized by the presence of an attacker and is called a flooding attack, and it causes noise in wireless communication by sending flooding packets as well as exhausts energy.

6.2 Limitations of the study

The considerations of the previous chapters have revealed that this thesis has adequately achieved its aims and objectives: the establishment of a novel cooperative IIDPS to use when DDoS attack in a wireless environment. However, a number of limitations and challenges were encountered during the study and they are listed here for future reference:

- We define a model for generating DDoS attack based upon the poison distribution function. This is due to lack of real DDoS attack dataset in wireless sensor network. The purpose of this section is to analyse the quantitative behaviour of attacks in the UDP protocol layer. To generate an attack, a random function was employed, which selected subject nodes from each cluster to attack. The selected nodes adjusted their functions to send flooding packets to the cluster head displays the attack strategy (See Algorithm 4.1).
- The hybrid machine learning algorithm cannot be used to cope with fast network changes as well as attack's behaviour fluctuations. However, as a remarkable advantage, the use of long-term statistical data leads to more robust methods. In

addition, as the temporary limitation is given by the measurement periods, there is plenty of time to apply complex optimization methods, which can further improve wireless sensor network performance. Thus, the term that refers to this kind of tasks is off-line tuning methods.

- A particular agent can be nominated the task of perceiving abnormal traffic flow in the network. Though, the single agent's action suffers from some weaknesses. The disadvantages are: 1) the attackers may be exterior the observation range of the detector node; 2) a large set of normal and abnormal patterns will have to be stored and processed by the detector node, for individually victim node of the network; 3) the lack of multiple decision making strategy on the detector node implies that the attack traffic flow may overwhelm the detector node itself, and thus disrupt the entire detection process.

6.3 Future Work

The set of compromised nodes participating in attacks may designate to send requests to the victim nodes at regular intervals of time by staying well below the attack detection threshold in WSN. In reality, the intensity of attack traffic may be constituted of malicious packets intending to cause damage to target nodes over a longer period of time. This type of an attack will lead to a gradual decline in resources of the target nodes in WSN. We can refer to this attack as a slow poisoning attack. A future direction of work can involve detection of such attacks in addition to detection of high traffic intensity attacks, addressed in this thesis.

The proposed attack detection scheme does detection of attacks that culminate from higher orders of incoming traffic within a single time epoch (See Section 5.2.5), without correlating traffic behaviour from previous time epochs. This work can be extended to incorporate correlation between time epochs, for attack detection purposes.

In addition, the length of the time epoch is static post-initialization. Variable time epoch lengths, based on analysis of real-time network traffic, are another possible future direction of research.

By defining incentives for cooperation and disincentives for fraudulent behaviour, it has been determined that repeated interaction sustains cooperation, builds confidence and enhances reputation, something additionally offered by Game-FQL. Game theory-based Fuzzy Q-learning (Game-FQL), as a mechanism in IDPS, is an invaluable tool for increasingly securing next-generation complex heterogeneous computing and networking environments against sophisticated attacks and attackers, beyond what is encountered today. A future initiative is to extend the proposed Game-FQL mechanism by incorporating data from various attack types and sources to further enhance its decision making capabilities in order to thwart existing or new attacks. Also as part of future research work on complementing Game-FQL, studying a network evolutionary algorithm, such as the imperialist competitive algorithm, is considered of utmost importance.

References

Abadeh, M. S., et al. (2007). Intrusion detection using a fuzzy genetics-based learning algorithm. Journal of Network and Computer Applications, 30(1), 414-428.

Abraham, A., et al. (2007). D-SCIDS: distributed soft computing intrusion detection system. Journal of Network and Computer Applications, 30(1), 81-98.

Agah, A., et al. (2007). Preventing DoS attacks in wireless sensor networks: A repeated game theory approach. International Journal of Network Security, 5(2), 145-153.

Agah, A., et al. (2004). Intrusion detection in sensor networks: A non-cooperative game approach. The Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04), pp. 343 - 346.

Ahmadabadi, M. N., et al. (2001). Cooperative Q-learning: the knowledge sharing issue. Advanced Robotics, 15(8), 815-832.

Akyildiz, I. F., et al. (2002). Wireless sensor networks: a survey. J. Comput. Netw, 38(4), 393-422.

Alpadin, E. (2010). Introduction to Machine Learning. Cambridge, Massachusetts: MIT Press.

Andersen, K. T., et al. (2009). Experiments with online reinforcement learning in real-time strategy games. Applied Artificial Intelligence, 23(9), 855-871.

Anderson D, Lunt TF, Javitz H, et al. (1995). Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES): Mountain View, California: SRI International, Computer Science Laboratory.

Anderson, J. A., et al. (1995). An introduction to neural network. Cambridge, Massachusetts:MIT Press.

Anstee, D., Bussiere, D., Sockrider, G., & Morales, C. (2012). Worldwide Infrastructure Security Report. Infrastructure Security Report,8.

Anuar, N. B., et al. (2012). A Response Strategy Model for Intrusion Response Systems. In Information Security and Privacy Research, Springer, 573-578.

Anuar N. B.,et al. (2014), A response selection model for intrusion response systems: Response Strategy Model (RSM), Security Comm. Networks,7,1815–1830

Aydın, M. A., et al. (2009). A hybrid intrusion detection system design for computer network security. Computers & Electrical Engineering, 35(3), 517-526.

Baig, Z. A., et al. (2010). Fuzzy Logic-Based Decision Making for Detecting Distributed Node Exhaustion Attacks in Wireless Sensor Networks. Proceedings of the Second International Conference on In Future Networks, 2010. ICFN'10, pp. 185-189.

Balajinath, B., et al. (2001). Intrusion detection through learning behavior model. Computer Communications, 24(12), 1202-1212.

Bankovic, Z. F., et al. (2011). Improving security in WMNs with reputation systems and self-organizing maps. Journal of Network and Computer Applications, 34(2), 455-463.

Barto, A. G. (1998). Reinforcement learning: An introduction. Cambridge, Massachusetts: MIT press.

Bicakci, K., et al. (2009). Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks. Computer Standards & Interfaces, 31(5), 931-941.

Bivens, A., et al. (2002). Network-based intrusion detection using neural networks. Intelligent Engineering Systems through Artificial Neural Networks, 12(1), 579-584.

Blasco, J., et al. (2010). Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming. Proceedings of the ARES'10 International Conference on Availability, Reliability, and Security, pp. 327-332.

Bokareva, T., et al. (2006). Wireless sensor networks for battlefield surveillance. Proceedings of Land Warfare Conference (LWC), Brisbane. pp. 1-11.

Bridges, S. M., et al. (2000). Fuzzy data mining and genetic algorithms applied to intrusion detection. Proceedings of 12th Annual Canadian Information Technology Security Symposium, pp. 109-122.

Buckley, J. J., et al. (1994). Fuzzy genetic algorithm and applications. Fuzzy Sets and Systems, 61(2), 129-136.

Cannady, J. (1998). Artificial neural networks for misuse detection. Proceeding of the National Information Systems Security Conference (NISSC'98), Arlington, VA. pp. 368-381.

Chavan, S., et al. (2004). Adaptive neuro-fuzzy intrusion detection systems. The Proceeding of International Conference on Information Technology: Coding and Computing, ITCC 2004, 1(70-74).

Chen, B., et al. (2002). Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. J. WIREL. NETW, 8(5), 481-494.

Curiac, D.-I., et al. (2012). Ensemble based sensing anomaly detection in wireless sensor networks. Expert Systems with Applications, 39(10), 9087-9096.

Da Silva, A. P. R., et al. (2005). Decentralized intrusion detection in wireless sensor networks. Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, pp. 16-23.

DARPA. (2012). Counter-Sniper Program (C-sniper). Retrieved 19 October, 2012, from http://www.darpa.mil/Our_Work/STO/Programs/Counter-Sniper_Program_(C-Sniper).aspx.

Dasgupta, D., et al. (2005). CIDS: An agent-based intrusion detection system. Computers & Security, 24(5), 387-398.

Dataset, C. (2006). Sensor network dataset for enhancing CSMA MAC protocol. from http://crawdad.cs.dartmouth.edu/meta.php?name=columbia/ecsma

Davis, J. J., et al. (2011). Data preprocessing for anomaly based network intrusion detection: A review. Computers & Security, 30(6), 353-375.

Debar, H., et al. (1992). A neural network component for an intrusion detection system. In Proceedings of IEEE Computer Society on Research in Security and Privacy, Oakland, CA. pp. 240-250.

Denning, D. E. (1987). An Intrusion-Detection Model. J. IEEE. T. Software. Eng, SE-13(2), 222-232.

Devarakonda, N., et al. (2012). Integrated Bayes Network and Hidden Markov Model for Host based IDS. International Journal of Computer Applications, 41(20), 45.

Dickerson, J. E., et al. (2001). Fuzzy intrusion detection. In Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS), Atlanta,GA. pp. 1506-1510.

Doelitzscher, F., et al. (2012). An agent based business aware incident detection system for cloud environments. Journal of Cloud Computing: Advances, Systems and Applications, 1(1), 9.

Dorigo, M., et al. (2010). Ant colony optimization. Encyclopedia of Machine Learning (36-39) : Heidelberg :Springer.

Dutkevych, T., et al. (2007). Real-Time Intrusion Prevention and Anomaly Analyze System for Corporate Networks. The 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 599-602.

Eik Loo, C., et al. (2006). Intrusion Detection for Routing Attacks in Sensor Networks. International Journal of Distributed Sensor Networks, 2(4), 313-332.

Elias, J., et al. (2011). Non-cooperative spectrum access in cognitive radio networks: A game theoretical model. Computer Networks, 55(17), 3832-3846.
Engelbrecht, A. P. (2007). Computational intelligence: an introduction. New York: Wiley.
Fisch, D., et al. (2012). Learning from others: Exchange of classification rules in intelligent distributed systems. Artificial Intelligence, 187, 90-114.
Fragkiadakis, A., et al. (2012). Design and performance evaluation of a lightweight wireless early warning intrusion detection prototype. EURASIP Journal on Wireless Communications and Networking, 2012(1), 1-18.
Fuchsberger, A. (2005). Intrusion Detection Systems and Intrusion Prevention Systems. J. Information Security Technical Report (ISTR), 10(3), 134-139.
Fullér, R. (2000). Introduction to neuro-fuzzy systems, Edittion 2. Heidelberg: Springer.
Gaing, Z.-L. (2004). A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Transactions Energy Conversion, 19(2), 384-391.
Garcia-Teodoro, P., et al. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18-28.
Ghosal, A., et al. (2013). Intrusion Detection in Wireless Sensor Networks: Issues, Challenges and Approaches. Wireless Networks and Security, Heidelberg: Springer. pp. 329-367.
Glorennec, P. Y. (1994). Fuzzy Q-learning and dynamical fuzzy Q-learning. Proceedings of the Third IEEE Conference on Fuzzy Systems, pp. 474-479.
Gomez, J., et al. (2002). Evolving fuzzy classifiers for intrusion detection. Proceedings of the 2002 IEEE Workshop on Information Assurance, 6 (3), 321-323. New York: IEEE Computer Press.
Group, V. (2004). UCB/LBNL/VINT network simulator NS (version 2). Retrieved in http://www.isi.edu/nsnam/ns.
Gu, G., et al. (2006). Measuring intrusion detection capability: an information-theoretic approach. Proceedings of the 2006 ACM Symposium on Information, computer and communications security, Taipei, Taiwan. pp. 90-101.
Gupta, I., et al. (2005). Cluster-head election using fuzzy logic for wireless sensor networks. Proceedings of the 3rd Annual Conference on Communication Networks and Services Research. pp. 255-260.
Hanson, M. A., et al. (2009). Body Area Sensor Networks: Challenges and Opportunities. J. IEEE. Computer, 42(1), 58-65.
Herrero, Á., et al. (2009). MOVIH-IDS: A mobile-visualization hybrid intrusion detection system. Neurocomputing, 72(13-15), 2775-2784.
Hofmeyr, S. A., et al. (2000). Architecture for an Artificial Immune System. Evolutionary Computation, 8(4), 443-473.
Huang, J.-Y., et al. (2013). Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining. Information Sciences, 231(1), 32-44.
Idris, N. B., et al. (2005). Artificial Intelligence Techniques Applied to Intrusion Detection. 2005 Annual IEEE in INDICON. pp. 52-55.
Intel Berkeley Research lab. (2004). Available From http://db.csail.mit.edu/labdata/labdata.html
Jianhui, L., et al. (2008). A fast fuzzy set intrusion detection model. International Symposium on Knowledge Acquisition and Modeling KAM'08. pp. 601-605.
Jungwon, K., et al. (2001). Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator. Proceedings of the 2001 Congress on Evolutionary Computation, 2 (pp.1244-1252).

Kapitanova, K., et al. (2012). Using fuzzy logic for robust event detection in wireless sensor networks. Ad Hoc Networks, 10(4), 709-722.
Kaspersky, E. (2013). The Cybercrime Ecosystem. Kaspersky Lab. Retrieved from www.kaspersky.com.
KDD. (1999). KDD Cup 1999 Data. Retrieved 1st April, 2012, from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
Khalil, I., et al. (2012). CTAC: Control traffic tunneling attacks' countermeasures in mobile wireless networks. Computer Networks, 56(14), 3300-3317.
Khalil, I., et al. (2010). UnMask: Utilizing neighbor monitoring for attack mitigation in multihop wireless sensor networks. Ad Hoc Networks, 8(2), 148-164.
Khan, S. A., et al. (2012). Application of fuzzy inference systems to detection of faults in wireless sensor networks. Neurocomputing, 94(0), 111-120.
Khanna, R., et al. (2009). Reduced Complexity Intrusion Detection in Sensor Networks Using Genetic Algorithm. IEEE International Conference on Communications, Dresden. pp. 1 - 5.
Kolias, C., et al. (2011). Swarm intelligence in intrusion detection: A survey. Computers & Security, 30(8), 625-642.
Kulkarni, R. V., et al. (2011). Computational intelligence in wireless sensor networks: A survey. IEEE Communications Surveys & Tutorials, 13(1), 68-96.
Law, Y. W., et al. (2005). Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, Alexandria, VA, USA. pp. 76 - 88.
Law, Y. W., et al. (2009). Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. ACM Trans. Sen. Netw., 5(1), 1-38.
Lee, C.-C. (1990). Fuzzy logic in control systems: fuzzy logic controller. IEEE Transactions on Systems, Man and Cybernetics, 20(2), 404-418.
León, O., et al. (2011). Towards a Cooperative Intrusion Detection System for Cognitive Radio Networks Networking Workshops. Lecture Notes in Computer Science, Heidelberg: Springer. 6827, pp. 231-242.
Li, B., et al. (2009). Using mobile agents to recover from node and database compromise in path-based DoS attacks in wireless sensor networks. Journal of Network and Computer Applications, 32(2), 377-387.
Li, N., et al. (2009). Privacy preservation in wireless sensor networks: A state-of-the-art survey. Ad Hoc Networks, 7(8), 1501-1514.
Li, W., et al. (2012). Security Through Collaboration and Trust in MANETs. Mobile Networks and Applications, 17(3), 342-352.
Li, Y., et al. (2012). An efficient intrusion detection system based on support vector machines and gradually feature removal method. Expert Systems with Applications, 39(1), 424-430.
Li, Z., et al. (2014). Increasing mapping based hidden Markov model for dynamic process monitoring and diagnosis. Expert Systems with Applications, 41(2), 744-751.
Li, Z., et al. (2012). Node localization through physical layer network coding: Bootstrap, security, and accuracy. Ad Hoc Networks, 10(7), 1267-1277.
Liang, Q., et al. (2005). Event detection in wireless sensor networks using fuzzy logic system.Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety. pp. 52-55.
Liu, L. (2008). System and Method for Distributed Denial of Service Identification and Prevention: Google Patents.
Lockhart, A. (2005). Snort wireless. Retrieved from the web. 2012, from http://snort-wireless.org

Ma, J., et al. (2007). SAID: A Self-Adaptive Intrusion Detection System in Wireless Sensor Networks Information Security Applications, Lecture Notes in Computer Science, pp. 60-73

Maggi, F., et al. (2009). Reducing false positives in anomaly detectors through fuzzy alert aggregation. Information Fusion, 10(4), 300-311.

Mariano, C. E., et al. (2001). DQL: A new updating strategy for reinforcement learning based on Q-learning, Machine Learning. Lecture Notes in Computer Science. pp 324-335

McGregory, S. (2013). Preparing for the next DDoS attack. Network Security, 2013(5), 5-6.

Mirkovic, J., et al. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39-53.

Misra, S., et al. (2011). Reputation-based role assignment for role-based access control in wireless sensor networks. Computer Communications, 34(3), 281-294.

Mohajerani, M., et al. (2003). NFIDS: a neuro-fuzzy intrusion detection system. Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems. PP. 348-351.

Mona Taghavi., et al. (2012). Taxonomy and Proposed Architecture of Intrusion Detection and Prevention Systems for Cloud Computing. The 4th International Symposium on Cyberspace Safety and Security (CSS 2012), Deakin University, Melbourne, Australia. pp. 441-458.

Mosqueira-Rey, E., et al. (2007). A Misuse Detection Agent for Intrusion Detection in a Multi-agent Architecture Agent and Multi-Agent Systems:Technologies and Applications. Lecture Notes in Computer Science, pp. 466-475.

Muñoz, P., et al. (2013). Optimization of load balancing using fuzzy Q-Learning for next generation wireless networks. Expert Systems with Applications, 40(4), 984-994.

Naserian, M., et al. (2009). Game theoretic approach in routing protocol for wireless ad hoc networks. Ad Hoc Networks, 7(3), 569-578.

Nash, J. F. (1950). Equilibrium points in n-person games. Proceedings of the national academy of sciences, 36(1), 48-49.

NSL-KDD. (2009). The NSL-KDD Data Set. 2009, from http://nsl.cs.unb.ca/NSL-KDD

Patcha, A., et al. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12), 3448-3470.

Pathan, A.-S. K. (2014). The State of the Art in Intrusion Prevention and Detection. Boca Raton, Florida: CRC Press.

Phillips, M., et al. (2010). Breath biomarkers of active pulmonary tuberculosis. Tuberculosis, 90(2), 145-151.

Ponomarchuk, Y., et al. (2010). Intrusion detection based on traffic analysis and fuzzy inference system in wireless sensor networks. Journal of Convergence 1(1). 35-42.

Potyrailo, R. A., et al. (2012). Wireless sensors and sensor networks for homeland security applications. J. Trac-Trend Anal Chem, 40(1) 133-145.

Precup, R.-E., et al. (2011). A survey on industrial applications of fuzzy control. Computers in Industry, 62(3), 213-226.

Qazi, S., et al. (2013). Securing DSR against wormhole attacks in multirate ad hoc networks. Journal of Network and Computer Applications, 36(2), 582-592.

Qiming, H., et al. (2000). Using reinforcement learning for pro-active network fault management. Proceedings of the International Conference on Communication Technology, Beijing. pp.515-521.

Ramachandran, C., et al. (2008). FORK: A novel two-pronged strategy for an agent-based intrusion detection scheme in ad-hoc networks. Computer Communications, 31(16), 3855-3869.
Renjit, J. A., et al. (2011). Multi-Agent-Based Anomaly Intrusion Detection. Information Security Journal: A Global Perspective, 20(4-5), 185-193.
Rolla, V. G., et al. (2013). A reinforcement learning-based routing for delay tolerant networks. Engineering Applications of Artificial Intelligence, 26(10), 2243-2250.
Russell, S. J., et al. (1995). Artificial intelligence: a modern approach. New Jersey: Prentice hall Englewood Cliffs.
Schaffer, P., et al. (2012). Secure and reliable clustering in wireless sensor networks: A critical survey. Computer Networks, 56(11), 2726-2741.
Sevil Sen, J. A. C. (2011). Evolutionary computation techniques for intrusion detection in mobile ad hoc networks. Computer Networks, 55(15), 3441–3457.
Shen, S., et al. (2011). Signaling game based strategy of intrusion detection in wireless sensor networks. Computers & Mathematics with Applications, 62(6), 2404-2416.
Sherif, J. S., et al. (2002). Intrusion detection: systems and models. Proceedings of Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 115-115.
Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. Proceedings of the 2001 Congress on Evolutionary Computation, pp. 81-86.
Shoham, Y., et al. (2009). Multiagent systems: Algorithmic, game-theoretic, and logical foundations. England: Cambridge University Press.
Stafrace, S. K., et al. (2010). Military tactics in agent-based sinkhole attack detection for wireless ad hoc networks. Computer Communications, 33(5), 619-638.
Steinberg, L. A., et al. (2005). Method and system for reducing false alarms in network fault management systems: Google Patents.
Sutton, R. S., et al. (1998). Reinforcement learning: An introduction. England : Cambridge Univ Press.
Takagi, T., et al. (1985). Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man and Cybernetics, 1(116-132).
Tong, W., et al. (2009). A Detection Method for Routing Attacks of Wireless Sensor Network Based on Fuzzy C-means Clustering. Proceeding of Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin. pp. 445-449.
Toosi, A. N., et al. (2007). A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Computer Communications, 30(10), 2201-2212.
Tsai, C.-F., et al. (2009). Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000.
Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. Machine Learning, 16(3), 185-202.
Vakili, G., et al. (2011). Coordination of cooperation policies in a peer-to-peer system using swarm-based RL. Journal of Network and Computer Applications. 35(2), 713-722.
Wang, B., et al. (2006). Local detection of selfish routing behavior in ad hoc networks. Journal of Interconnection Networks, 7(01), 133-145.
Wang, S.-S., et al. (2011). An Integrated Intrusion Detection System for Cluster-based Wireless Sensor Networks. Expert Systems with Applications, 38(12), 15234-15243.

Wang, Y. T., et al. (2012). ComSen: A Detection System for Identifying Compromised Nodes in Wireless Sensor Networks. The Sixth International Conference on Emerging Security Information, Systems and Technologies. pp. 148-156.

Wooldridge, M. (2009). An introduction to multiagent systems. New York: John Wiley & Sons.

Wu, S. X., et al. (2010). The use of computational intelligence in intrusion detection systems: A review. Applied Soft Computing, 10(1), 1-35.

Xing, K., et al. (2010). Attacks and Countermeasures in Sensor Networks: A Survey. In S. C. H. Huang, et al. (Eds.), Network Security, pp. 251-272.

Xu, X. (2010). Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies. Applied Soft Computing, 10(3), 859-867.

Xu, X., et al. (2007). Defending DDoS Attacks Using Hidden Markov Models and Cooperative Reinforcement Learning Intelligence and Security Informatics. Lecture Notes in Computer Science, pp. 196-207.

Xu, X., et al. (2005). A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls Advances in Intelligent Computing. Lecture Notes in Computer Science, pp. 995-1003.

Yan Li, W. J. (2012). The method of network intrusion detection based on the neural network GCBP algorithm. Processing of the International Conference on Computer Science and Information (CSIP). pp.1082-1086.

Ye, N. (2000). A markov chain model of temporal behavior for anomaly detection. Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, pp. 166- 169.

Zadeh, L. A. (1965). Fuzzy sets. Information and Control,8(3), 338-353.

Zadeh, L. A. (1994). Soft computing and fuzzy logic. IEEE Software, 11(6), 48-56.

Zhang, Y., et al. (2000). Intrusion detection in wireless ad-hoc networks. Proceedings of the 6th annual international conference on Mobile computing and networking, Boston, Massachusetts, United States. pp. 275-283.

Zhang, Z., et al. (2001). HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. Proceedings of the IEEE Workshop on Information Assurance and Security, pp. 85-90.

List of Publications

Published Journal Articles

1. **Shahaboddin Shamshirband**, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ahmed Patel, An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique, Engineering Applications of Artificial Intelligence, Volume 26, Issue 9, Pages 2105–2127, October 2013.
2. **Shahaboddin Shamshirband**, Ahmed Patel, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ajith Abraham, Cooperative Game Theoretic Approach using Fuzzy Q-learning for Detecting and Preventing Intrusions in Wireless Sensor Networks, Engineering Applications of Artificial Intelligence. Volume 32, Pages 228–241, June 2014.
3. **Shahaboddin Shamshirband**, Nor Badrul Anuar, Miss Laiha Mat Kiah, Dalibor Petković, Sanjay Misra, Co-FAIS: Cooperative Fuzzy Artificial Immune System for Detecting Intrusion in Wireless Sensor Networks, Journal of Network and Computer Applications. Volume 42, Pages 102–117, June 2014.
4. **Shahaboddin Shamshirband**, Nor Badrul Anuar, Miss Laiha Mat Kiah, Sanjay Misra, Anomaly Detection using Fuzzy Q-learning Algorithm in Wireless Network, Acta Polytechnica Hungarica, Volume 11, Issue 8, Pages 5-28, November 2014.
5. **Shahaboddin Shamshirband**, Amineh Amini, Nor Badrul Anuar, Miss Laiha Mat Kiah, Teh Ying Wah, Steven Furnell, D-FICCA: A Density-based Fuzzy Imperialist Competitive Clustering Algorithm for Intrusion Detection in Wireless Sensor Networks, Measurement, Volume 55, Pages 212–226, September 2014.
6. **Shahaboddin Shamshirband**, Babak Daghighi, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ahmed Patel, Ajith Abraham. Co-FQL: Anomaly Detection Using Cooperative Fuzzy Q-learning in Network, - Journal of Intelligent and Fuzzy Systems, Doi: 10.3233/IFS-141419.

Appendix

```
#include"QLearning.h"
#include <time.h>
#include <string.h>
#include <math.h>
#include <windows.h>
int Total = 0;
int temp[10000][10000];
********** FUNCTION   Name:  Get Node Details**********
int get_node_count() {
	printf("get_node_count( )");
	int node_cnt;
	printf("\n\n");
	printf("=================================================\n");
	printf(">>>>>>>>>>>>     Get the Node Details <<<<<<<<<<< \n");
	printf("=================================================\n");
	printf("\n\nEnter the Total No of Node to be Created :");
	scanf("%d", &node_cnt);
	printf("\n\nEnter the Selfish Node Count          :");
	scanf("%d", &snode_cnt);
	printf("\n\n");
	printf("=================================================\n");
	return node_cnt;
}
**********FUNCTION   Name:  splitstr**********
splitstr(char * full_str, char arr[][64], char * str) {
	char *record = NULL;
	int count = 0;
	// Take the Records and store into the Array
	record = strtok(full_str, str);
	while (record != NULL) {
		//fmt_str[count] = malloc( strlen( record ) + 1 );
		strcpy(arr[count++], record);
		record = strtok(NULL, str);
	}
	return count;
}
**********       Name:  validateip **********
validateip(char * ip, char ipset[][10]) {
	char *record = NULL;
	int count = 0;
	// Take the Records and store into the Array
	record = strtok(ip, ".");
	while (record != NULL) {
		//fmt_str[count] = malloc( strlen( record ) + 1 );
		if (strcmp(record, "XXX") == 0)
			strcpy(ipset[count++], "-1");
		else
			strcpy(ipset[count++], record);
		record = strtok(NULL, ".");
	}
	if (count == 4)
		return TRUE;
	return FALSE;
}
```

```
**********FUNCTION Name: is_star**********
int is_star(char * str) {
        if (strcmp(str, "*") == 0 || strcmp(str, "XXX") == 0)
                return -1;
        else
                return atoi(str);
}
**********Name: validate_split_ip**********
int validate_split_ip(char * ip, struct IP_Set * ipaddr) {
        char arr[10][64];

        if (splitstr(ip, arr, ".") == 4) {
                printf(" ARR : %s - %s - %s - %s \n ", arr[0], arr[1], arr[2], arr[3]);
                ipaddr->Ip1 = is_star(arr[0]);
                ipaddr->Ip2 = is_star(arr[1]);
                ipaddr->Ip3 = is_star(arr[2]);
                ipaddr->Ip4 = is_star(arr[0]);
                return TRUE;
        }
        return FALSE;
}
**********FUNCTION Name: get_rules **********
int get_rules() {
        printf("\n\n");
        /*      printf( "================================================\n");
        printf( ">>>>>>>>>>>>  Define the Rules       <<<<<<<<<<< \n");
        printf( "================================================\n");*/
        printf("1.Reading the KDD dataset....");
        printf("\n2.Initializing Expert system....\n");
        // Validate TCP or UDP
        do {
                printf(" Enter the Protocal ( 1. TCP , 2.UDP ) : ");
                scanf("%d", &ql_rules.protocal);
                if (ql_rules.protocal > 0 && ql_rules.protocal < 3)
                        break;
                else
                        printf("Error : Sorry Enter the Valid protocal\n");
        } while (TRUE);
sinknode_cnt = 1;
printf("Initializing Qstates ... ");
}
**********FUNCTION Name:
print_nodes**********
int print_nodes() {
        int i, j;
        char node_type[25];
        int randval;
        printf("\n\n");
        printf("Total No of Packets : %d \n", noofnode);
        Total = noofnode;
        printf("Total No of Abnormal Packets : %d \n", snode_cnt);
                for (i = 0; i < noofnode; i++) {
                                if (node_details[i].node_type == 1) {
                                strcpy(node_type, "Normal");
                } else {
                        strcpy(node_type, "Selfish");
                        randval = random_in_range(0, sinknode_cnt);
                        sink_node_vals[randval].ids[sink_node_vals[randval].count++] = i;
                }

                if ((strcmp(node_type, "Normal") == 0)
                                && ((strcmp(node_details[i].state, "d8") == 0)
                                                || (strcmp(node_details[i].state, "d9") == 0)
```

```
                                                || (strcmp(node_details[i].state, "d10") == 0))) {
                        true_pos++;
                }
                if ((strcmp(node_type, "Normal") == 0)
                                && ((strcmp(node_details[i].state, "d8") != 0)
                                                && (strcmp(node_details[i].state, "d9") != 0)
                                                && (strcmp(node_details[i].state, "d10") != 0))) {
                        true_neg++;
                }
                if ((strcmp(node_type, "Selfish") == 0)
                                && ((strcmp(node_details[i].state, "d8") != 0)
                                                && (strcmp(node_details[i].state, "d9") != 0)
                                                && (strcmp(node_details[i].state, "d10") != 0))) {
                        false_neg++;
                }
                if ((strcmp(node_type, "Selfish") == 0)
                                && ((strcmp(node_details[i].state, "d8") == 0)
                                                || (strcmp(node_details[i].state, "d9") == 0)
                                                || (strcmp(node_details[i].state, "d10") == 0))) {
                        false_pos++;
                }
                printf( "Node Number : %d Type : %s  IP Address : %03d.%03d.%03d.%03d  Bytes
Transfered : %d Count : %d State : %s \n" , node_details[i].Node_id , node_type ,
node_details[i].ipaddr.Ip1 ,  node_details[i].ipaddr.Ip2 ,  node_details[i].ipaddr.Ip3 ,
node_details[i].ipaddr.Ip4  , node_details[i].src_bytes , node_details[i].count ,node_details[i].state);}*/
        }
int val;
for (i = 0; i < sinknode_cnt; i++) {
                for (j = 0; j < sink_node_vals[i].count; j++) {
                        val = sink_node_vals[i].ids[j];
Transfered : %d Count : %d \n" , node_details[val].Node_id , node_type , node_details[val].ipaddr.Ip1 ,
node_details[val].ipaddr.Ip2 ,  node_details[val].ipaddr.Ip3 ,  node_details[val].ipaddr.Ip4  ,
node_details[val].src_bytes , node_details[val].count );
         (strcmp(node_details[i].state,"d9")==0) || (strcmp(node_details[i].state,"d10")==0)){
         printf( "Node Number : %d    count_Level : %s    Buffer_level : %s     Level : %s   State : %s\n"
, node_details[val].Node_id ,   node_details[val].count_level,
node_details[val].buffer_level,node_details[val].level, node_details[val].state);}
        }
}
int print_selfish_node() {
        printf("3.Anomaly nodes detected ...");
        int val;
        int i, j;
        for (i = 0; i < sinknode_cnt; i++) {
                printf("Sink Node index : %d \n", i);
                for (j = 0; j < sink_node_vals[i].count; j++) {
                        val = sink_node_vals[i].ids[j];
                        printf("Node Number : %d   Level : %s \n",
                                        node_details[val].Node_id, node_details[val].level);
                }

        }

}
* ******************************      Name:  setipaddr ******************************
int setipaddr(struct IP_Set *ipaddr) {
        ipaddr->Ip1 = random_in_range(0, 255);
        ipaddr->Ip2 = random_in_range(0, 255);
        ipaddr->Ip3 = random_in_range(0, 255);
        ipaddr->Ip4 = random_in_range(0, 255);
        return TRUE;
}
******************************Name:  create_rule_ip******************************
```

```
int create_rule_ip(struct IP_Set *ipaddr) {
	//printf( " %d=%d=%d=%d \n", ql_rules.ipaddr.Ip1 , ql_rules.ipaddr.Ip2 , ql_rules.ipaddr.Ip3 ,
ql_rules.ipaddr.Ip4 ) ;
	ipaddr->Ip2 = random_in_range(0, 255);
	ipaddr->Ip3 = random_in_range(0, 255);
	ipaddr->Ip4 = random_in_range(0, 255);
	return TRUE;
}
******************************Name:  create_nodes*****************************
int create_nodes() {
	int i;
	printf("\n\n");
	/*	printf( "================================================\n");
	 printf( ">>>>>  Going to generate the Random Details <<<<<< \n");
	 printf( "================================================\n");*/
	for (i = 0; i < noofnode; i++) {
		node_details[i].Node_id = i;
		create_rule_ip(&node_details[i].ipaddr);

	}
	int nodenum;
	for (i = 0; i < snode_cnt; i++) {
		nodenum = random_in_range(0, noofnode);
		setipaddr(&node_details[nodenum].ipaddr);
	}

}
******************************Name:  random_in_range*****************************
int random_in_range(unsigned int min, unsigned int max) {
	int base_random = rand();
	if (RAND_MAX == base_random)
		return random_in_range(min, max);
	if (base_random < RAND_MAX - remainder) {
		return min + base_random / bucket;
	} else {
		return random_in_range(min, max);
	}
}
******************************Name:  display*****************************
display(int start, int end) {
	int i, j;
	for (i = start; i <= end; i++) {
		for (j = start; j <= end; j++)
			printf("%4d", adj[i][j]);
		  printf("\n");
	}

}
display_con(int start, int end) {
	int i, j;
	printf("Displaying connection matrix\n");
	for (i = start; i < end; i++) {
		for (j = start; j < end; j++)
			printf("%4d", con_mat[i][j]);
		printf("\n");
	}

}
******************************Name:  formated_display*****************************
formated_display() {
	int i, j;
	for (i = 0; i <= n; i++) {
		if (i == 0) {
```

```
                puts("");
                for (j = 0; j <= n; j++)
                        (j == 0) ? printf("     |") : printf(" %3d |", j);
                puts("");
                while (j--)
                        printf("-----+");
        } else
                for (j = 0; j <= n; j++) {
                        if (j == 0)
                                printf(" %3d |", i);
                        else
                                printf(" %3d |", adj[i][j]);
                }
        printf("\n");
    }

}
```

Name: findpath

```
int findpath(int s, int d, int path[MAX], int *sdist) {
    struct node state[MAX];
    int i, min, count = 0, current, newdist, u, v;
    *sdist = 0;
    for (i = 1; i <= n; i++) {
        state[i].predecessor = 0;
        state[i].dist = infinity;
        state[i].status = TEMP;
    }
    state[s].predecessor = 0;
    state[s].dist = 0;
    state[s].status = PERM;
    current = s;
    while (current != d) {
        for (i = 1; i <= n; i++) {
            if (adj[current][i] > 0 && state[i].status == TEMP) {
                newdist = state[current].dist + adj[current][i];
                if (newdist < state[i].dist) {
                    state[i].predecessor = current;
                    state[i].dist = newdist;
                }
            }
        }
        min = infinity;
        current = 0;
        for (i = 1; i <= n; i++) {
            if (state[i].status == TEMP && state[i].dist < min) {
                min = state[i].dist;
                current = i;
            }
        }
        if (current == 0)
            return 0;
        state[current].status = PERM;
    }
    while (current != 0) {
        count++;
        path[count] = current;
        current = state[current].predecessor;
    }
    for (i = count; i > 1; i--) {
        u = path[i];
        v = path[i - 1];
        *sdist += adj[u][v];
    }
```

```
	return (count);
}
node_split() //Splits packets into nodes
{
	int i, j, k, l, split_node[100], packet_node[100];
	for (i = 0; i < 10000; i++) {
		node_sp[i].node_id = 0;
	}
	for (i = 0; i < n; i++) {
		if (strcmp(node_details[i].state, "d1") == 0) {
			node_sp[1].node_id++;
			strcpy(node_sp[1].node_state1, "d1");
		}
		if (strcmp(node_details[i].state, "d2") == 0) {
			node_sp[2].node_id++;
			strcpy(node_sp[2].node_state1, "d2");
		}
		if (strcmp(node_details[i].state, "d3") == 0) {
			node_sp[3].node_id++;
			strcpy(node_sp[3].node_state1, "d3");
		}
		if (strcmp(node_details[i].state, "d4") == 0) {
			node_sp[4].node_id++;
			strcpy(node_sp[4].node_state1, "d4");
		}
		if (strcmp(node_details[i].state, "d5") == 0) {
			node_sp[5].node_id++;
			strcpy(node_sp[5].node_state1, "d5");
		}
		if (strcmp(node_details[i].state, "d6") == 0) {
			node_sp[6].node_id++;
			strcpy(node_sp[6].node_state1, "d6");
		}
		if (strcmp(node_details[i].state, "d7") == 0) {
			node_sp[7].node_id++;
			strcpy(node_sp[7].node_state1, "d7");
		}
		if (strcmp(node_details[i].state, "d8") == 0) {
			node_sp[8].node_id++;
			strcpy(node_sp[8].node_state1, "d8");
		}
		if (strcmp(node_details[i].state, "d9") == 0) {
			node_sp[9].node_id++;
			strcpy(node_sp[9].node_state1, "d9");
		}
		if (strcmp(node_details[i].state, "d10") == 0) {
			node_sp[10].node_id++;
			strcpy(node_sp[10].node_state1, "d10");
		}
	}
	k = 0;
	total_noofpack = 0;
	goal_node_number = 0;
	for (i = 1; i <= 10; i++) {

		if (node_sp[i].node_id < 30) {
			split_node[i] = 1;
			k = k + 1;
			strcpy(node_sp[k].node_state, node_sp[i].node_state1);

				if (((i == 8) || (i == 9) || (i == 10)) ){total_noofpack =
total_noofpack+1;
				goal_node_index[goal_node_number] = i;
```

```
                        goal_node_number = goal_node_number+1;
                }
        } else if (node_sp[i].node_id >= 30 && node_sp[i].node_id < 50) {
                split_node[i] = 2;
                k = k + 2;
                strcpy(node_sp[k-1].node_state, node_sp[i].node_state1);
                strcpy(node_sp[k].node_state, node_sp[i].node_state1);

                if (((i == 8) || (i == 9) || (i == 10)) ){total_noofpack = total_noofpack+1;
                        goal_node_index[goal_node_number] = k-1;
                        goal_node_number = goal_node_number+1;
                        goal_node_index[goal_node_number] = k;
                        goal_node_number = goal_node_number+1;
                }
        } else {
                split_node[i] = ceil(node_sp[i].node_id / 50);
                for (l = k+1; l<=(k+split_node[i]);l++)
                {
                        strcpy(node_sp[l].node_state, node_sp[i].node_state1);
                        if (((i == 8) || (i == 9) || (i == 10)) ){
                        goal_node_index[goal_node_number] = l;
                        goal_node_number = goal_node_number+1; }
                }
                k = k + split_node[i];
                if (((i == 8) || (i == 9) || (i == 10)) ){total_noofpack = total_noofpack+1;}
        }

    }
    noofpack = k;
    for (i =1 ; i<=10; i++){
    printf("Printing node state \n n%d = %s ==> nodecount =
%d\n",i,node_sp[i].node_state1,node_sp[i].node_id);
}   printf("Total No of Node Count : %d \n", noofpack);
}
******************************   Name:  create_graph ******************************
create_graph() {
    int i, j, max_edges, origin, destin, wt;
    // Define Rules
    get_rules();

    //  Read Data From Kdd set
    find_minmax();
    n = read_kdd_dataset();
    noofnode = n;
        print_nodes();
    create_nodes();
    max_edges = n * (n - 1);
    node_split();
    for (i = 0; i < noofpack; i++) {
        for (j = 0; j < noofpack; j++) {
                con_mat[i][j] = (rand() % (2 - 0)) + 0;
        }
    }
    for (i = 0; i < noofpack; i++)
        for (j = 0; j < noofpack; j++) {
            origin = i;
            destin = j;
            if ((i==j) || (con_mat[i][j] == 1))
            {if ((strcmp(node_sp[j+1].node_state1,"d8")==0) ||
(strcmp(node_sp[j+1].node_state1,"d9")==0) || (strcmp(node_sp[j+1].node_state1,"d10")==0))
                {
                        wt = 100;
                }
```

```
                                        else if ((i != j) || (con_mat[i][j] == 1))
                                        {
                                                wt = 0;
                                        }
                                        else
                                        {
                                                wt = -1;
                                        }
                                }
                                else
                                {
                                        wt = -1;
                                }

        if (origin > noofpack || destin > noofpack || origin < 0 || destin < 0) {
                printf("\nInvalid edge!\n");
                i--;
        } else
                adj[origin][destin] = wt;
}
}
****************************        Name:  find_minmax ****************************
find_minmax() {
FILE *fp;
char *filename = "kddset.data";
fp = fopen(filename, "r");
char *line;
line = (char *) malloc(1024);
if (fp == NULL) {
        perror("Error : Opening kdd data set file");
        return 0;
}
char arr[1000][64];
while (fgets(line, 1024, fp)) {
        splitstr(line, arr, ",");
        if (time_min > atoi(arr[0]))
                time_min = atoi(arr[0]);
        if (time_max < atoi(arr[0]))
                time_max = atoi(arr[0]);
        if (bs_min > atoi(arr[5]))
                bs_min = atoi(arr[5]);
        if (bs_max < atoi(arr[5]))
                bs_max = atoi(arr[5]);
        if (count_min > atoi(arr[22]))
                count_min = atoi(arr[22]);
        if (count_max < atoi(arr[22]))
                count_max = atoi(arr[22]);
}
time_avg = (time_max - (time_min)) / 3;
bs_avg = (bs_max - (bs_min)) / 3;
count_avg = (count_max - (count_min)) / 3;
}
****************************        Name:  read_kdd_dataset ****************************
int read_kdd_dataset() {
FILE *fp;
char *filename = "kddset.data";
fp = fopen(filename, "r");
char *line;
int count = 0;
snode_cnt = 0;
line = (char *) malloc(1024);
if (fp == NULL) {
        perror("Error : Opening kdd data set file");
```

```
		return 0;
}
char arr[50][64];
while (fgets(line, 1024, fp)) {
		splitstr(line, arr, ",");
		node_details[count].protocal = (strcmp(arr[1], "tcp") == 0) ? 1 : 2;
		// Check the node details
		if (ql_rules.protocal == node_details[count].protocal) {
			create_rule_ip(&node_details[count].ipaddr);
			node_details[count].node_type = 1;
		} else {
			setipaddr(&node_details[count].ipaddr);
			node_details[count].node_type = 2;
			snode_cnt++;
		}
		node_details[count].Node_id = count;
		node_details[count].src_bytes = atoi(arr[4]);
		node_details[count].dst_bytes = atoi(arr[5]);
		node_details[count].count = atoi(arr[22]);
		node_details[count].time = atoi(arr[0]);
		if (atoi(arr[0]) >= 0 && atoi(arr[0]) <= (time_min+time_avg))
			strcpy(node_details[count].level, "Low");
		else if (atoi(arr[0]) > time_avg && atoi(arr[0]) <= ((time_min)+(2 * time_avg)))
			strcpy(node_details[count].level, "Medium");
		else if (atoi(arr[0]) > ((time_min)+(2 * time_avg)))
			strcpy(node_details[count].level, "High");
		else
			strcpy(node_details[count].level, "High");

		if (atoi(arr[22]) >= 0 && atoi(arr[22]) <= (count_min+count_avg))
			strcpy(node_details[count].count_level, "Low");
		else if (atoi(arr[22]) > count_avg && atoi(arr[22]) <= ((count_min)+(2 * count_avg)))
			strcpy(node_details[count].count_level, "Medium");
		else if (atoi(arr[22]) > ((count_min)+(2 * count_avg)))
			strcpy(node_details[count].count_level, "High");
		else
			strcpy(node_details[count].count_level, "High");

		if (atoi(arr[5]) >= 0 && atoi(arr[5]) <= (bs_min+bs_avg))
			strcpy(node_details[count].buffer_level, "Low");
		else if (atoi(arr[5]) > bs_avg && atoi(arr[5]) <= ((bs_min)+(2 * bs_avg)))
			strcpy(node_details[count].buffer_level, "Medium");
		else if (atoi(arr[5]) > ((bs_min)+(2 * bs_avg)))
			strcpy(node_details[count].buffer_level, "High");
		else
			strcpy(node_details[count].buffer_level, "High");
		if ((strcmp(node_details[count].count_level, "Low") == 0
				&& strcmp(node_details[count].buffer_level, "Medium") == 0
				&& strcmp(node_details[count].level, "High") == 0)
				|| (strcmp(node_details[count].count_level, "Low") == 0
						&& strcmp(node_details[count].buffer_level, "High") == 0
						&& strcmp(node_details[count].level, "Low") == 0)
				|| (strcmp(node_details[count].count_level, "Low") == 0
						&& strcmp(node_details[count].buffer_level, "High") == 0
						&& strcmp(node_details[count].level, "Medium") == 0))
			strcpy(node_details[count].state, "d1");
		else if ((strcmp(node_details[count].count_level, "Low") == 0
				&& strcmp(node_details[count].buffer_level, "Medium") == 0
				&& strcmp(node_details[count].level, "Low") == 0)
				|| (strcmp(node_details[count].count_level, "Low") == 0
						&& strcmp(node_details[count].buffer_level, "Medium") ==
0
						&& strcmp(node_details[count].level, "Medium") == 0)
```

```
                || (strcmp(node_details[count].count_level, "Low") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "High") == 0))
                strcpy(node_details[count].state, "d2");
        else if ((strcmp(node_details[count].count_level, "Medium") == 0
                && strcmp(node_details[count].buffer_level, "High") == 0
                && strcmp(node_details[count].level, "Low") == 0)
                || (strcmp(node_details[count].count_level, "Medium") == 0
                                && strcmp(node_details[count].buffer_level, "High") == 0
                                && strcmp(node_details[count].level, "Medium") == 0))
                strcpy(node_details[count].state, "d3");
        else if ((strcmp(node_details[count].count_level, "Low") == 0
                && strcmp(node_details[count].buffer_level, "Low") == 0
                && strcmp(node_details[count].level, "Medium") == 0)
                || (strcmp(node_details[count].count_level, "Low") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "Low") == 0))

                strcpy(node_details[count].state, "d4");
        else if ((strcmp(node_details[count].count_level, "Medium") == 0
                && strcmp(node_details[count].buffer_level, "Low") == 0
                && strcmp(node_details[count].level, "Medium") == 0)
                || (strcmp(node_details[count].count_level, "Medium") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "Low") == 0)
                || (strcmp(node_details[count].count_level, "Medium") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "High") == 0)
                || (strcmp(node_details[count].count_level, "Medium") == 0
                                && strcmp(node_details[count].buffer_level, "Medium") ==
0
                                && strcmp(node_details[count].level, "Low") == 0)
                || (strcmp(node_details[count].count_level, "Medium") == 0
                                && strcmp(node_details[count].buffer_level, "Medium") ==
0
                                && strcmp(node_details[count].level, "Medium") == 0))
                strcpy(node_details[count].state, "d5");
        else if ((strcmp(node_details[count].count_level, "High") == 0
                && strcmp(node_details[count].buffer_level, "Medium") == 0
                && strcmp(node_details[count].level, "Medium") == 0)
                || (strcmp(node_details[count].count_level, "High") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "Low") == 0)
                || (strcmp(node_details[count].count_level, "High") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
                                && strcmp(node_details[count].level, "Medium") == 0)
                || (strcmp(node_details[count].count_level, "High") == 0
                                && strcmp(node_details[count].buffer_level, "Medium") ==
0
                                && strcmp(node_details[count].level, "Low") == 0))
                strcpy(node_details[count].state, "d6");
        else if (strcmp(node_details[count].count_level, "Medium") == 0
                && strcmp(node_details[count].buffer_level, "Medium") == 0
                && strcmp(node_details[count].level, "High") == 0)
                strcpy(node_details[count].state, "d7");
        else if ((strcmp(node_details[count].count_level, "High") == 0
                && strcmp(node_details[count].buffer_level, "High") == 0
                && strcmp(node_details[count].level, "Low") == 0)
                || (strcmp(node_details[count].count_level, "Low") == 0
                                && strcmp(node_details[count].buffer_level, "High") == 0
                                && strcmp(node_details[count].level, "High") == 0)
                || (strcmp(node_details[count].count_level, "High") == 0
                                && strcmp(node_details[count].buffer_level, "Low") == 0
```

```
                                        && strcmp(node_details[count].level, "High") == 0))
                strcpy(node_details[count].state, "d8");
        else if ((strcmp(node_details[count].count_level, "Medium") == 0
                        && strcmp(node_details[count].buffer_level, "High") == 0
                        && strcmp(node_details[count].level, "High") == 0)
                        || (strcmp(node_details[count].count_level, "High") == 0
                                        && strcmp(node_details[count].buffer_level, "Medium") ==
0
                                        && strcmp(node_details[count].level, "High") == 0)
                        || (strcmp(node_details[count].count_level, "High") == 0
                                        && strcmp(node_details[count].buffer_level, "High") == 0
                                        && strcmp(node_details[count].level, "Medium") == 0))
                strcpy(node_details[count].state, "d9");
        else if (strcmp(node_details[count].count_level, "High") == 0
                        && strcmp(node_details[count].buffer_level, "High") == 0
                        && strcmp(node_details[count].level, "High") == 0)
                strcpy(node_details[count].state, "d10");

        count++;
}
fclose(fp);
return count;
}
int isGoal(int node) {
char myState[10];
strcpy(myState, node_details[node].state);
if (strcmp(myState, "d8") == 0 || strcmp(myState, "d9") == 0
                || strcmp(myState, "d10") == 0)
        return 1;
else
        return 0;
}
int isGoal_1(int i,int j) {

if (adj[i][j]==100)
        return 1;
else
        return 0;
}
****************************** Name:  main ******************************
void d1(int start, int end) {
int i, j;
printf("\n");
for (i = start; i <= end; i++) {
        for (j = start; j <= end; j++)
                printf("%d\t", cur[i][j]);
        printf("\n");
}
}
Ql(int start, int end, int sink1) {
int sink_no, i, j, k, l, max = 0, cur_state = 2, next_state = 0, loop = 0, ini = 1,
                isnxt_state = 1, epoc = 0, check_last,goal_count = 0,goal_check = 1,goal_node,

        ac_i,ac_j,ac_reach,ac_start,ac_col,ac_find[noofpack],ac_i1,ac_notreach,ac_new,ac_oreach;
        double divide;
        int m,n,acc_count,acc_total;
        int total_acc=0;
        sink_no = sink1;
        time_t now;
        time_t now1;
        int *second_array = (int*) malloc(noofpack*sizeof(int));
        int *goal_array = (int*) malloc(noofpack*sizeof(int));
        int state_count = 0, initial_state = -1, second_state = -1;
```

```
		for (i = 0; i<noofpack;i++)
		{
		second_array[i] = -1;
		ac_find[i] = 0;
		}
		{
		for (j=1;j<=n;j++)
		 {
				adj[i][j] = -1;
				cur[i][j] = 0; } }
 adj[1][5]=0; adj[2][4]=0; adj[2][6]=100; adj[3][4]=0; adj[4][2]=0; adj[4][3]=0; adj[4][5]=0; adj[5][1]=0;
adj[5][4]=0; adj[5][6]=100; adj[6][2]=0; adj[6][5]=0; adj[6][6]=100;
printf("\n");

if (sink_no == -1) {
		time(&now);
		printf(" ql starts %s\n", ctime(&now));
		sink_no = sinknode_cnt + 1;
}
FILE *f1, *f2, *f3;
		f1 = fopen("Expertness.txt", "wt");
		//fprintf(f1, "%s,%s\n", "Epoch_number", "Expertness");
		f2 = fopen("Accuracy.txt", "wt");
		f3 = fopen("Time.txt", "wt");
		//fprintf(f2, "%s,%s\n", "Epoch_number", "Accuracy");
		fclose(f2);
		fclose(f3);
		fclose(f1);
// initializing Q matrix(cur)
for (i = 0; i<noofpack;i++){
		for (j = 0; j < noofpack; j++){
			cur[i][j] =0;}}
			display_con(0, noofpack);
			printf("\nDisplaying Reward matrix\n");
			display(0,noofpack-1);
			if (loop = 1)			d1(0, noofpack-1);
		for (m = 0 ; m < noofpack ; m++)
			{
				for (n = 0 ; n < noofpack ; n++)
				{
					temp[m][n]  = 0;			}			}

// Epoc for starts...
		ac_oreach = 0;
		for (loop = 1; loop < 500; loop++) {
			goal_node = 0;
			goal_count = 0;
			goal_check = 1;
			time(&now);

		// Node for loop starts....
		for(j=0 ; j< noofpack; j++)
		{
			initial_state = j;
			state_count = 0;
			// second state for loop starts
			for (k = 0; k < noofpack; k++)
			{
				//printf("j = %d \n k = %d \n con_mat[][] = %d\n",j,k,con_mat[j][k]);
				//getch();
				if (con_mat[j][k] > 0)
				{
```

```
                        second_array[state_count] = k;
                        //printf("second array: %d\t",second_array[state_count]);
                        state_count++;
                }
        // if connection not available
        if (state_count == 0)
                continue;
                int index = (rand() % ((state_count+1) - 0)) + 0;
                second_state =  second_array[index];
                // Q Function loop starts
                max = 0;
                for ( l = 0 ; l < noofpack ; l++)
                {       if ( con_mat[k][l] > 0 )
                        {
                                goal_check = 1;
                                if (cur[k][l] > max)
                                {
                                max = cur[k][l];                        }
}
                        }
                goal_check = 1;
                if (adj[initial_state][second_state] == 100)
                {
                   for (goal_node =0 ; goal_node<goal_count;goal_node++)
                        {
                        if (goal_array[goal_node] == second_state)
                        {
                                goal_check = goal_check + 1;
                        }
                }
                if (goal_check == 1)
                {
                        goal_array[goal_count] = second_state;
                        goal_count++;
                }
        }

        if (((adj[initial_state][second_state] % 2) == 1) || (adj[initial_state][second_state] == -
1)){
        cur[initial_state][second_state] = adj[initial_state][second_state] + (0.8 * max)+1;
}
        else{
        }
}
if (epoc != -1) {
        expert_agent.sum_rewards[sink_no] = 0;
        expert_agent.total_rewards[sink_no] = 0;
        expert_agent.expertness[sink_no] = 0;
        for (i = start; i <= end; i++) {
                for (j = start; j <= end; j++) {
                        expert_agent.sum_rewards[sink_no] =
                                        expert_agent.sum_rewards[sink_no] + cur[i][j];
                        if (expert_agent.sum_rewards[sink_no] < 0) {
                        }
                        expert_agent.total_rewards[sink_no]++;
                }
        }
        expert_agent.expertness[sink_no] = expert_agent.sum_rewards[sink_no]
total_acc = 0;
for (m = 0 ; m < noofpack ; m++)
{
        for (n = 0 ; n < noofpack ; n++)
        {
```

```
                        if (temp[m][n] != cur[m][n])
                        {
                                total_acc = total_acc+1;                        }       }
        }
        for (m = 0 ; m < noofpack ; m++)
        {
                for (n = 0 ; n < noofpack ; n++)
                {
                        temp[m][n] = cur[m][n];    }
        }
  ac_reach = 0;
        ac_notreach = 0;
        ac_start = -1;
void main() {
        int i, j; int source, dest; int path[MAX];int energy[MAX]; int shortdist, count;int total_energy =
        0;
        double weight_sink = 0;
        create_graph();
        int enery_val[MAX];
        int total_time;
        printf("\n");
        printf("3.Q Learning starts....\n");
        printf("****** Complete QLearning ******\n");
        Ql(1, noofpack, -1);
        int split = Total / sinknode_cnt;
        expert_agent.total_expert = 0;
                while (1) {
                        scanf("%d", &source);
                        printf("\nEnter destination node(0 to quit) : ");
                        scanf("%d", &dest);}
```