جامعة الشرق الأوسط
MIDDLE EAST UNIVERSITY

# Enhanced Solutions for Misuse Network Intrusion Detection System using SGA and SSGA

# الحلول المحسنة لنظام كشف التطفل باستخدام الخوارزمية الجينية البسيطة والخوارزمية الجينية الحالة المستقرة

**By:**

**Sabah Abdulazeez Jebur**

**Supervisor:**

**Dr. Hebah H. O. Nasereddin**

**A Thesis Submitted In Partial Fulfillment of the Requirements of the**

**Master Degree in Computer Information Systems**

**Faculty of Information Technology**

**Middle East University**

**June, 2015**

## Authorization Statement

I, Sabah Abdulazeez Jebur, authorize Middle East University to supply a copy of my thesis to libraries, establishments or individuals.

Signature: [illegible]

Date: 3/6/2015

انا صباح عبدالعزيز جبر افوض جامعة الشرق الاوسط بتزويد نسخة من رسالتي للمكتبات او المؤسسات او الهيئات او الافراد عند طلبها.

التوقيع: [illegible]

انا صباح عبدالعزيز جبر افوض جامعة الشرق الاوسط بتزويد نسخة من رسالتي للمكتبات او المؤسسات او الهيئات او الافراد عند طلبها.

التوقيع:

## Examination Committee Decision

This is to certify that the thesis entitled "**Enhanced Solutions for Misuse Network Intrusion Detection System using SGA and SSGA**" was successfully defended and approved on 3/6/2015

| **Examination Committee Member** | **Signature** |
|---|---|
| **1. Dr. Suhail Owais** | [illegible] |
| **2. Dr. Mohammed Alhussainy** | [illegible] |
| **3. Dr. Hebah Nasereddin** | [illegible] |

## *Acknowledgments*

***All credit and success is due to the Merciful Allah always and forever.***

***I would like to extend my deepest gratitude to my supervisor Dr. Hebah Nasereddin for all of her helpful comments and guidance's throughout this work.***

***I am grateful to my parents and family who always provided help and support throughout my academic life and career.***

***I would like to express my heart-felting gratitude to my wife for her love, concern, support, encouragement and inspiration throughout my research work and life.***

***I am thankful to my brother and true friend Ward Ahmed for his illuminating views and support throughout the work of this thesis.***

***Last but not least, I would like to express my sincere appreciation to the Iraqi Government for giving me the opportunity to complete my Masters Degree in Jordan.***

# *Dedication*

***To my father and mother***

***To my beloved Wife***

***To my brothers and sisters***

***No words can make me express my gratitude and love.***

***To the martyrs' souls of Iraq.***

# Table of Contents

# List of Tables

# List of Figures

## List of Abbreviations

| | |
|---|---|
| AF | average of fitness value |
| BFSS | Best Feature Set Selection |
| BTR | Binary Tournament Replacement |
| DARPA | Defense Advanced Research Projects Agency |
| DoS | Denial of Service |
| DR | Detection Rate |
| FFSA | Forward Feature Selection  Algorithms |
| FPR | False Positive Rate |
| GA | Genetic Algorithm |
| GP | Genetic Programming |
| HIDS | Host-based Intrusion Detection system |
| ID | Intrusion Detection |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| KDD99 | Knowledge Discovery in Databases 1999 |
| LR | Logistic Regression |
| MMIFS | Modified Mutual Information Feature Selection |
| NIDS | Network-based Intrusion detection system |
| OWASP | Open Web Application Security Project |
| R2L | Remote to Local Attack |
| RWS | Roulette Wheel Selection |
| SGA | Simple Genetic Algorithm |
| SGA-IDS | Simple genetic algorithm based Intrusion Detection system |
| SSGA | Steady State Genetic Algorithm |
| SSGA-IDS | Steady State genetic algorithm based Intrusion Detection system |
| SUS | Stochastic Universal Sampling |
| SVM | Support Vector Machine |
| SYN | Synchronization |
| TCP | Transmission Control Protocol |
| TTR | Triple Tournament Replacement |
| U2R | User to Root Attack |

# Enhanced Solutions for Misuse Network Intrusion Detection System using SGA and SSGA

**By:**
**Sabah Abdulazeez Jebur**

**Supervisor:**

**Dr. Hebah H. O. Nasereddin**

## Abstract

One of the most widely acknowledged purposes of using the Internet is data transfer; it is an essential way of communicating personal and sensitive data. Therefore, the need for protecting such data against hackers and intruders is at most importance. Many security systems were built for this purpose; Intrusion Detection Systems (IDS) are one of those systems. The main function of Intrusion Detection System is to monitor the incoming connections and detect attacks.

The purpose of this thesis is to verify the power of Simple Genetic Algorithm (SGA) versus Steady State Genetic Algorithm (SSGA) in intrusion detection field. This is achieved by developing two models of IDS. In the first model, the Simple Genetic Algorithm was used to build IDS (SGA based IDS), while in the second model; Steady State Genetic Algorithm was used to build IDS (SSGA based IDS). The evaluations and the experiments were performed using the NSL-KDD intrusion detection dataset.

The experimental results demonstrated that performing an IDS using SGA gives higher performance results than using SSGA according to the value of Detection rate (DR) where it achieved an average of DR equal to 88.5%, while SSGA based IDS achieved an average of DR equal to 72.53%. Also the number of the new generated rules using SGA is more than the number of the new generated rules using SSGA, despite that, the training time for SGA experiments is shorter than the training time for SSGA. On other hand, SSGA based IDS produced an average of False Positive Rate (FPR) equal to 4.66% which is considered relatively better than SGA based IDS that produced an average of FPR equal to 5.21%.

***Key words:*** Intrusion Detection System, Simple Genetic Algorithm, Steady State Genetic Algorithm.

# الحلول المحسنة لنظام كشف التطفل باستخدام الخوارزمية الجينية البسيطة والخوارزمية الجينية الحالة المستقرة

**اعداد:**
**صباح عبدالعزيز جبر**

**اشراف:**
**د. هبه حسن ناصرالدين**

## الخلاصة

احد اهم اسباب استخدام الانترنت هو نقل البيانات حيث يعتبر الوسيلة الرئيسية لنقل البيانات الشخصية والحساسة. بالتالي فإن حماية هذه البيانات من المتطفلين والدخلاء هي حاجة بالغة الاهمية. العديد من انظمة الحماية تم بناءها لهذا الغرض، وتعتبر انظمة كشف التطفل (IDS) هي احد هذه الانظمة. المهمة الاساسية لنظام كشف التطفل هي مراقبة الاتصالات الواردة والكشف عن الهجمات.

الغرض من هذه الدراسة هو للتحقق من قوة الخوارزمية الجينية البسيطة (SGA) مقارنة بالخوارزمية الجينية الحالة المستقرة (SSGA) في مجال كشف التطفل. تم تحقيق ذلك من خلال تطوير نموذجين من انظمة كشف التطفل. في النموذج الاول تم استخدام الخوارزمية الجينية البسيطة لبناء نظام كشف التطفل، بينما في النموذج الثاني تم استخدام الخوارزمية الجينية الحالة المستقرة لبناء نظام كشف التطفل. اجريت عمليات التقييم والتجارب باستخدام مجموعة بيانات تدعى (NSL-KDD).

أظهرت النتائج التجريبية أن تنفيذ نظام كشف التطفل باستخدام الخوارزمية الجينية البسيطة يعطي نتائج اداء اعلى من استخدام الخوارزمية الجينية الحالة المستقرة وفقا لقيمة معدل كشف التطفل حيث أنه حقق متوسط معدل كشف يساوي 88.5٪، في حين حقق نظام كشف التطفل باستخدام الخوارزمية الجينية الحالة المستقرة متوسط معدل كشف يساوي 72.53٪. أيضا عدد القواعد الجديدة المتولدة باستخدام الخوارزمية الجينية البسيطة هو أكثر من عدد القواعد الجديدة المتولدة باستخدام الخوارزمية الجينية الحالة المستقرة. على الرغم من ذلك، وقت التدريب المستغرق لتجارب الخوارزمية الجينية البسيطة كان أقصر من وقت التدريب للخوارزمية الجينية الحالة المستقرة. من جهه اخرى، حقق نظام كشف التطفل باستخدام الخوارزمية الجينية الحالة المستقرة معدل خطأ ايجابي يساوي 4.66٪ والذي يعتبر افضل نسبياً من نظام كشف التطفل باستخدام الخوارزمية الجينية البسيطة والذي حقق معدل خطأ ايجابي يساوي 5.21٪.

***الكلمات المفتاحية:*** نظام كشف التطفل، الخوارزمية الجينية البسيطة، الخوارزمية الجينية الحالة المستقرة.

# Chapter One

# Introduction

## 1.1 Preface

Nowadays, attacks on the computer resources are becoming an increasingly serious problem. Despite different techniques have been developed and deployed to protect computer systems against network attacks (anti-virus software, firewall, message encryption, secured network protocols, password protection), securing data communication over Internet and any other network is still under threat of intrusions and misuses (Abdullah, Abd-Alghafar, Salama, and Abd-Alhafez, 2009). Development of a smart and strong Intrusion Detection System (IDS) that monitors network traffic and provides a right alarm is a hot research area in Computer Security today.

According to Ojugo, Eboka, Okonta, Yoro, and Aghware, (2012), “Intrusion is the set of actions that attempts to compromise integrity, confidentiality or availability of network resources, while an intruder is any user or group of users who initiates such intrusive action”.

An IDS is a device or a software application that constantly monitors network or system activities for malicious activities or policy violations with the ability to analyze network traffic and recognize incoming network attack to produce reports to a management station (Chowdhary, Suri, and Bhutani, 2014).

The most existent IDSs face a number of challenges such as low detection rates and high false alarm rates, and therefore prevent authorized users from accessing the network resources, these problems occur because of the attacks' behavior and their intended similarities to normal behavior (Shaveta, Bhandari, and Saluja, 2014).

To overcome these problems in IDSs, intrusion detection must be implemented by using smart methods based on artificial intelligence techniques to detect attacks. One of the important approaches of artificial intelligence used to detect Intrusion is Genetic Algorithm (GA) (Shaveta, et al., 2014).

Ojugo, et al., (2012), showed that “GA is an effective heuristic search technique that finds the best solutions to an optimization task-inspired from biological, evolution process and natural genetics proposed by Holland in 1975”. They also reported that Functioning of GA starts with randomly generated population of chromosomes which are continuously evaluated via a fitness function. A population consists of set of chromosomes which represent the candidate solutions, where each chromosome has number of genes. Through various generations these chromosomes evolved and their quality gets improved. In every generation, three basic operators of genetic algorithm i.e. selection, crossover and mutation are applied to the population to gradually improve the quality of each chromosome. The new pool goes through the same process and continues until a termination condition is reached.

A number of researchers, as mention in chapter two, have studied the effects of Simple Genetic Algorithm (SGA) and Steady State Genetic Algorithm (SSGA) in variety of applications in order to compare their behavior and study their practicability. Duran and Xhafa, (2006) confirmed that SGA outperforms SSGA for Job Scheduling on Computational Grids, while Jones and Soule, (2006) showed that SSGA gives better results than SGA in “two peaks” problem. This thesis will verify the power of SGA versus SSGA in intrusion detection field.

## 1.2 Problem Statement

After the emergence of computer networks and especially the Internet, the communication of data and the exchange of information have become very easy and fast. The easy access and exchange of information over the network brings the question of how to ensure the security of the information that is stored or exchanged against attacks by the intruders.

Network Intrusion detection system (NIDS) is one of the key security components in today's networking environment. NIDS must be more efficient to detect intrusions with maximizing Detection Rate (DR) and minimizing False Positive Rate (FPR).

The main questions in this study are identified as follows:

- How to develop and apply Intrusion Detection System using Genetic Algorithm to protect data communications and computer networks against attacks by the intruders?
- Can the proposed models identify normal and abnormal behavior on network and if abnormal is detected, can find which type of attack it is with high DR and low FPR?
- By comparing between Simple Genetic Algorithm (SGA) and Steady State Genetic Algorithm (SSGA), what is the best form of Genetic Algorithm in intrusion detection field?

## 1.3 Contribution

The contribution of this research is to verify the power of Simple Genetic Algorithm (SGA) versus Steady State Genetic Algorithm (SSGA) in intrusion detection field, by calculating:

- Training time
- Number of new generated rules
- Detection Rate (DR)
- False Positive Rate (FPR)

## 1.4 Objectives

The main objective of this research is developing and applying an IDS using GA to detect and classify misuse intrusion according to their types (DOS, R2L, U2R and Probing) with high DR and low FPR in order to help computer systems to prepare for attacks and deal with them.

## 1.5 Limitations

- This research will detect and classify the incoming attacks according to four main categories of attacks (DOS, R2L, U2R and Probing), not sub types of these attacks.
- There are two categories of IDS: Network-based IDS and host-based IDS, this research will perform Network-based IDS.
- There are two approaches of intrusion detection: Misuse and Anomaly Detection; this research will deal with Misuse Detection.

## 1.6 Thesis Outline

This thesis consists of five chapters organized as follows:

**Chapter Two:** This chapter discusses in details each of Intrusion Detection system and its classifications, Genetic Algorithm and its elements and its categories. Also, it views NSL-KDD intrusion detection dataset which was used in this research and finally presents literature review and related works in the field of Intrusion Detection and Genetic Algorithm.

**Chapter Three:** The methodology of this research and the proposed models structure will be discussed in this chapter. It also presents the evaluation measurements used in this study.

**Chapter Four:** The implementation stages of the proposed models will be provided step by step in this chapter. Also presenting experimental results and providing a concise discussion for those results. Finally, those results are compared with other research results.

**Chapter Five:** This chapter presents the conclusions of this study and future work suggestions.

# Chapter Two

# Theoretical Background & Literature Review

## 2.1 Overview

This chapter discusses the concepts of computer security, Threats and Attacks, Intruders and Viruses, and Intrusion Detection. It also introduces the knowledge about Intrusion Detection system and its classifications. On the other side, this chapter explains the elements and categories of Genetic Algorithm, and its working mechanism. Also, it discusses the network intrusion detection dataset, NSL-KDD dataset, which was used in this research. Finally, this chapter introduces some related works in the field of intrusion detection using Genetic algorithm and other machine learning algorithms.

## 2.2 Computer Security

Computer Security is the process of protecting sensitive resources and their critical characteristics (Confidentiality, Integrity, and Availability) in a computer system by applying mechanisms access to protected resources (Stallings and Brown, 2008).

- **Confidentiality**

  "Assures that private or confidential information is not made available or disclosed to unauthorized individuals" (Stallings and Brown, 2008).

- **Data Integrity**

  "Assures that information and programs are changed only in a specified and authorized manner" (Stallings and Brown, 2008).

- **Availability**

  "Assures that systems work promptly and service is not denied to authorized users" (Stallings and Brown, 2008).

### 2.2.1 Threats & Attacks

Threat is a potential danger that might exploit a flaw or weakness in a system for violation of system's security policy and could cause harm, while an attack is a threat that is carried out by actions (Stallings and Brown, 2008).

### 2.2.2 Intruders & Viruses

Intruders and viruses are the most popular threats to security. Intruder is an unauthorized person who uses a computer system and penetrates the system’s access controls to exploit legitimate users' account, or a person who is authorized for such an access but misuses his or her privileges (Al-Zokari, 2008).

A virus is a written program that attaches itself to another program which could cause damage; it is loaded onto a computer (Al-Zokari, 2008).

## 2.3 Intrusion Detection (ID)

Intrusion Detection (ID) is "a security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner" (Stallings and Brown, 2008).

Firewalls, access controls, and authentication facilities play a major role in countering intrusions. ID is often used as another line of defense to monitor and analyze incoming connection in order to detect network intrusions by assuming that the intruder's behavior differs from the authorized user's behavior.

## 2.4 Intrusion Detection System (IDS)

An Intrusion Detection System (IDS) is defined by Chowdhary, et al., (2014) “a device or a software application that monitors network or system activities for malicious activities or policy violations and produces reports to a management station. Intrusion detection systems constantly monitor a given computer network for invasion or abnormal activity”.

The motivations of using IDS for intrusions is detected quickly and ejected from the system before causing any damage, it is an action to prevent intrusions. IDS also depends in its work on a collection of information about the behavior of the intruder to strengthen the intrusion prevention measures (Al-Zokari, 2008).

Aziz, (2014) reported IDS goals as follows:

- To detect a wide variety of intrusions; including intrusion from the inside and the outside as well as known and unknown attacks.
- Detect intrusions in a timely fashion.
- Present the analysis in a simple, easy - to- understand format.
- Be accurate, that is, achieve as a low false-alarm rate.

## 2.5 Intrusion Detection Systems Classifications

Intrusion Detection Systems can be classified into two categories on the basis of the detection approaches: Misuse-based IDS and Anomaly-based IDS. IDSs also can be divided into two groups depending on the location where they look for intrusive actions: Host-based IDS (HIDS) and Network-based IDS (NIDS) (Prasad and Borah, 2013).

### 2.5.1 Misuse-based Detection Technique

Misuse-based IDS looks for sequences of known events to identify attacks. It depends on a definite set of rules as attack patterns that can be used to detect the intrusion. The advantage of this technique lies within its capability to detect known attacks with detection rate higher than anomaly detection. The main disadvantage, it lacks the ability to detect the newly invented attacks as well as some variations of existing attacks. Misuse-based IDS are also called signature-based intrusion detection (Shiri, Shanmugam, and Idris, 2011).

### 2.5.2 Anomaly-based Detection Technique

Anomaly Detection techniques depends on collecting information about the behavior of authorized users over a period of time by analyzing incoming audit records to identify deviation from an average behavior. The advantages of this technique, it does not require a prior knowledge to detect attacks. The main disadvantages, the disability to identify the attack type and the high false positive rate. Anomaly detection attempts to quantify the usual or acceptable behavior and flag other irregular behavior as potentially intrusive (Shiri, et al.,2011).

### 2.5.3 Host-based Intrusion Detection System (HIDS)

A host-based IDS (HIDS) adds a specialized layer of security software to vulnerable or sensitive systems such as database servers and administrative systems.

The host-based IDS monitors the activity on the system in a variety of ways to detect suspicious behavior. It can detect both external and internal intrusions, something that is not possible either with network-based IDSs or firewalls (Stallings and Brown, 2008). HIDS performs the following:

- Using log files and/or the system's auditing agents as sources of data.
- Looking at the communications traffic in and out of a single computer.
- Checking the integrity of system files, and watching for suspicious processes, including changes to system files and user privileges.

One drawback for HIDS is that it weakens real-time response and the software should be installed on each computer on the network to be protected (Prasad and Borah, 2013).

### 2.5.4 Network-based Intrusion Detection System (NIDS)

A network-based IDS (NIDS) monitors network traffic for particular network segments or devices. The NIDS analyzes the traffic packets in real time or near to real time to identify suspicious activity. NIDS monitors packet traffic directed toward vulnerable computer systems on a network, unlike HIDS which monitors user and software activity on a host. It is most commonly deployed at a boundary between networks, such as in proximity to border firewalls or routers (Prasad and Borah, 2013).

Its disadvantage, it needs a more complex configuration and maintenance than the HIDS. It produces more FPR than the HIDSs, because it reads the network activity pattern in order to distinguish between normal and abnormal connections (Al-Zokari, 2008). Figure (2.1) depicts the locations of NIDS and HIDS on the network. This study will deal with Misuse Network IDS.

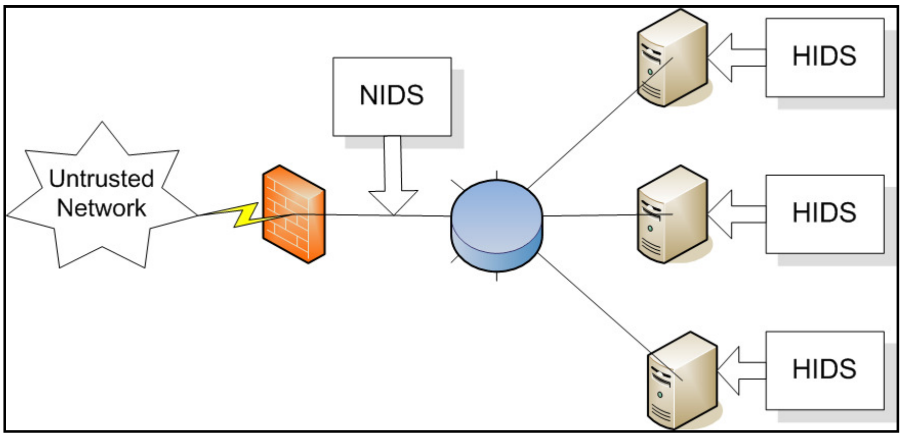


**Figure (2.1): The placement of the NIDS and HIDS in the Network**

## 2.6 Network Attacks

The main purpose of IDS is the detection of any potential network attacks. Different researches classify network attacks into four categories: Denial of Service (DoS), Probing, Remote to Local (R2L), and User to Root (U2R) (Hoque, Mukit, and Bikas, 2012), (Mukkamala, Sung, and Abraham, 2004), (Chou, Yen, Luo, 2008).

### 2.6.1 Denial of Service Attack (DoS)

A denial of service is a form of attack on the availability of some services. It prevents or impairs normal use of systems by making a computer resource unavailable to its users, for. For example the land attack; which sends a spoofed TCP SYN packet with the target host’s IP address as both source and destination. This causes the target to reply to itself continuously and crash.

### 2.6.2 Probing

Probing is a class of attacks in which an attacker scans a network of computers to gather information about the victim host in order to find weaknesses or vulnerabilities which are running on the target.

### 2.6.3 Remote to Local Attack (R2L)

A remote to local attack is a class of attacks in which an attacker sends packets to a victim machine over a network and then exposes the machine's weakness to gain local access as a user.

### 2.6.4 User to Root Attack (U2R)

User to root attacks are used to obtain unauthorized access to system administrator privileges by having a normal user account on the system and trying to exploit some vulnerability or weakness in order to gain root access to the system.

## 2.7 Genetic algorithm (GA)

As defined by Guo, Wang, and Han, (2010), "GA is the powerful stochastic algorithm which is applied in machine learning and optimization problems to solve complex problems. It is based on the principles of natural selection and natural genetics inspired by Darwin's principles in optimizing the chromosome population of candidate solutions. GA maintains a population of individuals and probabilistically modifies the population by some genetic operators such as selection, crossover and mutation, with the intent of seeking a near optimal solution to the problem".

GA produces new advanced solutions from current solutions. This is achieved by selecting solutions that have high fitness value to undergo information exchanges using genetic operators (Crossover and Mutation) in order to create more advanced chromosomes that consider advanced solutions (Aziz, 2014).

## 2.8 Working Mechanism of Genetic Algorithm

Adewumi, (2010) explain that GA starts with a population of individuals randomly sampled over the search space. Using fitness function, each individual is associated with a fitness value that reflects its quality. GA tries to improve the quality of the individuals by making the population evolve. This evolution is achieved using information exchanges between individuals in order to create new ones or modify the existing ones using genetic operators such selection, crossover and mutation. The selection operator helps with the exploitation of search space. While Crossover combines elements of individuals in the current generation to create individuals for the successive generations, it consists of exchanging genetic material between two selected single chromosomes. On the other hand, Mutation systematically changes elements of an individual in the current generation in order to introduce variety into the next generation. The main components of a GA are:

### 2.8.1 Population

At the start of a genetic algorithm, array of chromosomes will be randomly generated to cover the range of possible solutions (the search space), where each chromosome represents potential solution of the problem to be solved. The nature of the problem determines the population size (Jebur and Nasereddin, 2015).

### 2.8.2 Evaluation

The evaluation process is a very important measure to calculate the goodness of a chromosome. Fitness function is the heart of all Genetic Processes. It evaluates the performance of all chromosomes in the population, where a chromosome with high fitness value has a high probability to be selected in the selection stage (Dhak and Lade, 2012).

### 2.8.3 Encoding

Encoding is one of the main processes in GA to represent the data into some of the encoding formats. Various encoding methods have been created for particular problems to provide effective implementation of genetic algorithms. The encoding methods can be classified as follows (Dhak and Lade, 2012):

- **Binary Encoding:** This method converts the value of chromosome into binary value (0, 1).

e.g., chromosome =

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 |

- **Integer or Literal Permutation Encoding:** This method converts the value of chromosome into integer numbers.

e.g., chromosome =

| | | | | |
|---|---|---|---|---|
| 5 | 0 | 100 | 77 | 1220 |

- **Real Number Encoding:** in this method, the structure of genotype space is identical to that of the phenotype. Therefore, it is easy to form effective genetic. This method uses actual values of chromosome.

e.g., chromosome =

| | | | | |
|---|---|---|---|---|
| 0.5 | 1.6 | 100 | 0.77 | 5.5 |

### 2.8.4 Selection

In this process, multiple chromosomes are selected from the current population based on their fitness to produce successive generations. The better chromosomes have more chance of being selected and can be selected more than once to reproduce into the next generation (Jebur and Nasereddin, 2015). There are several schemes for the selection process:

- **Roulette Wheel Selection**: This selection method is easy to implement and resemble more to the nature. Usually a proportion of the wheel is assigned to each of the possible selection based on their fitness value. This could be achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated (Sharma, Singh, and Sharma, 2012).
- **Elitism Selection:** In this method, chromosomes are ordered descending according to their fitness function. Then each two arranged chromosomes are selected together. Here, GA will be applied either between strong chromosomes or between weak chromosomes (Alabsi, 2012).
- **Stochastic Universal Sampling (SUS)**: SUS is introduced by Baker (1987) for sampling individuals with lower variance. It is efficient requiring only a single pass over all the members of the population to select the mating population in contrast to Roulette Wheel Selection which needs passing n times on Roulette Wheel. SUS is doing by determining n points in the wheel. Then choose chromosomes that situated in front of determined points (French, 2012).
- **Rank Selection:** In this scheme, individuals are assigned selection probability based on their rank ordering in the current population instead of their fitness values. Where individuals are arranged descending according to their fitness value, then an individual with the highest fitness value takes the rank=1, the next takes the rank=2 and so on. Finally the Roulette Wheel can be used to choose the chromosomes. Calculate the new fitness value for each chromosome using the equation (2.1) (Aziz, 2014):

$$F = \max - (\max - \min) * \frac{rank - 1}{N_{pop} - 1} \tag{2.1}$$

Where 1<max<=2 & min = 2-max

- **Tournament Selection**: For this method, individuals are randomly chosen from the current population to compete in a tournament for selection. Commonly, binary tournament selection is used, whereby two individuals are chosen to compete and the individual with the higher fitness wins and is selected for mating (French, 2012).

### 2.8.5 Crossover Operator

"The crossover operator decides which genes of the parents should be swapped to generate the offspring" (Soon, Guan, On, Alfred, and Anthony, 2013). There are several crossover operators, such as:

- **Single-point crossover:** In the one-point crossover, a crossover point is randomly selected and the bit strings after that point are swapped between the two parents. Figure (2.2) shows the one-point crossover process.

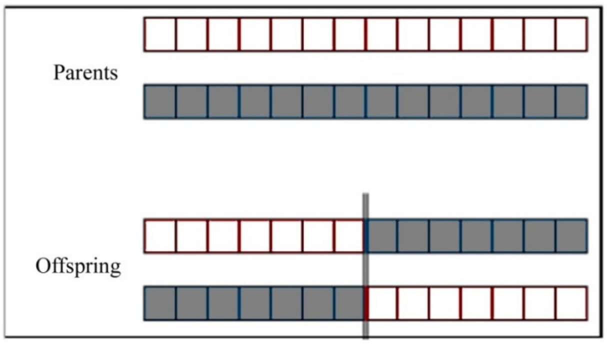


**Figure (2.2): One-point crossover**

- **Two-point crossover:** two bit positions are randomly selected and the bit strings between these two points are swapped. Figure (2.3) shows the two-point crossover operation.

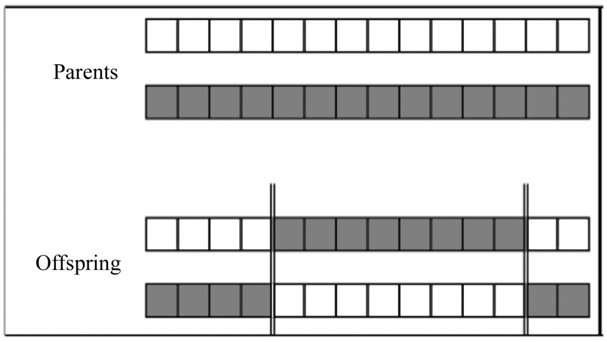


**Figure (2.3): Two-point crossover**

- **N-point crossover:** There are three types of N-point crossover point which are the Odd N-point crossover, the Even N-point crossover and the mixed N-point crossover. In the Odd N-point crossover, an odd random crossover point is chosen. The genes will be split based on the odd point and alternating between parents. Similar operation will be performed in the Even N-point crossover by splitting the genes based on the randomized even point and swapped between parents. The Mixed N-point crossover chooses n random crossover points, split along those points and alternating between parents. Figure (2.4) shows the N-point crossover process.

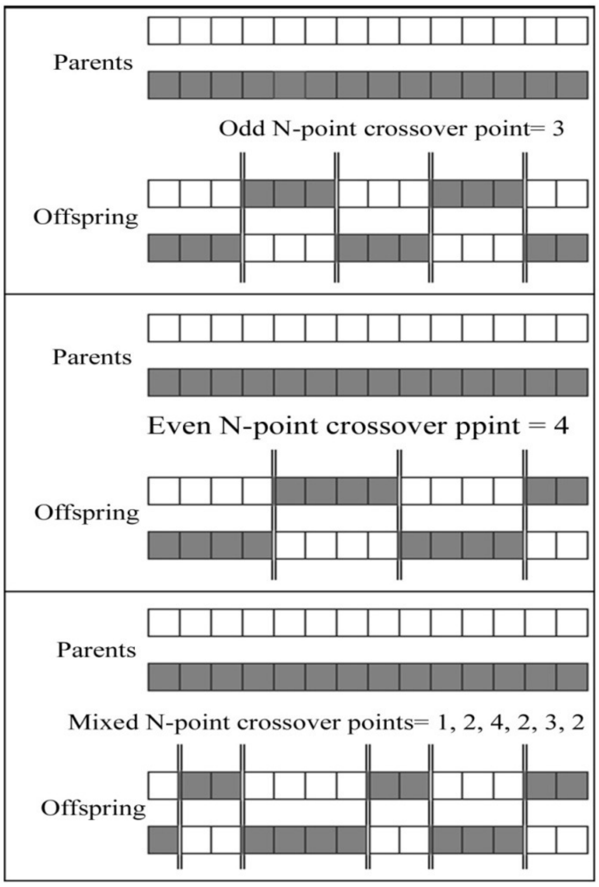

Parents
Odd N-point crossover point= 3
Offspring
Parents
Even N-point crossover ppint = 4
Offspring
Parents
Mixed N-point crossover points= 1, 2, 4, 2, 3, 2
Offspring

**Figure (2.4): N-point crossover**

- **Uniform crossover:** The uniform crossover uses a fixed mixing ratio between two parents. All genes have equal probability to be swapped. Thus, the offspring will have 50% of first parent's genes and another 50% from the second parent's genes. Figure (2.5) shows the Uniform crossover process.

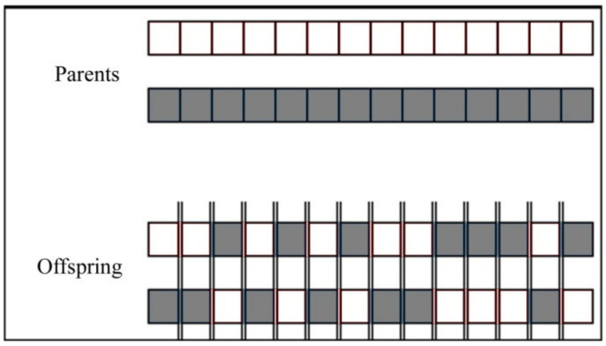


**Figure (2.5): Uniform crossover**

### 2.8.6 Mutation Operator

According to Hasan and Mustafa (2011), “Mutation operator is one of the GA operators that used to produce new chromosomes or modify some features of it depending on some small probability value. The objective of this operator is to prevent falling of all solutions in population into a local optimum of solved problem”. There are several types of mutation methods, such as:

- **Flip bit:** the value of a gene that was chosen randomly, to be equal to a random number of specific range.
- **Boundary:** an item is selected randomly and its value is replaced randomly either by the upper bound or the lower bound.
- **Inversion:** items between two randomly chosen points in the individual are reversed in order.
- **Insertion:** an item is taken at random and inserted randomly into another position in the sequence.
- **Displacement:** A randomly selected section of the individual is moved as a block to another location in the individual.
- **Uniform:** Replace the value of a chosen gene with a uniform random value selected between the user specified upper and lower bounds for that gene.

### 2.8.7 Replacement

"It is a process performed on the worst individuals to be replaced by better new individuals" (Naoum, Aziz, and Alabsi, 2014).There are two types of Replacement method:

- **Binary Tournament Replacement (BTR):** two individuals are chosen to compete and the individual with the higher fitness wins and is selected for mating.
- **Triple Tournament Replacement (TTR):** three individuals are chosen to compete and the individual with the highest fitness wins and is selected for mating.

### 2.8.8 Stopping criteria

This process defines the conditions under which the search process terminates. Typical stopping criteria include the following (French, 2012), (Kumar, Husian, Upreti, and Gupta, 2010):

- Maximum number of generations reached.
- Allocating budget (ex: time, money) reached.
- Successive iterations no longer produce better results.
- If there are no additional new solutions will be produced.
- Terminate if the optimal solution has been discovered.

## 2.9 GA Categories

Two categories of GA will be discussion in this section: Simple Genetic Algorithm (SGA) and Steady State Genetic Algorithm (SSGA).

### 2.9.1 Simple Genetic Algorithm (SGA)

Simple Genetic algorithm (SGA), also called as generational genetic algorithm, creates new chromosomes (offspring's) from current chromosomes (parents) using the genetic operators (Crossover and Mutation). These new chromosomes replaced previous chromosomes to form new population for the next generation, where all of the population undergoes transformation at each generation (Mehra, et al., 2012). Figure (2.6) shows the SGA structure.

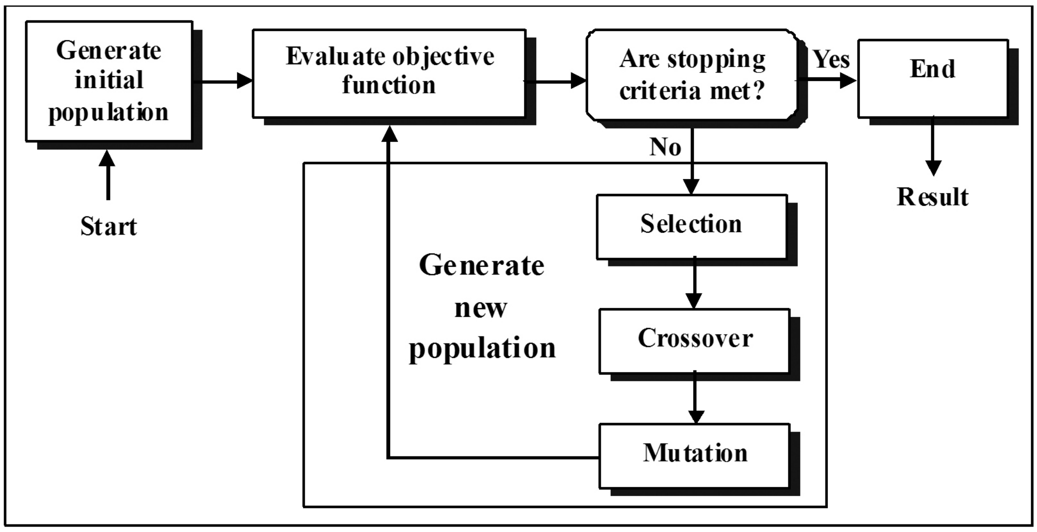

Generate initial population
Start
Evaluate objective function
Are stopping criteria met?
Yes
End
Result
No
Generate new population
Selection
Crossover
Mutation

**Figure (2.6): Simple Genetic Algorithm structure**

### 2.9.2 Steady State Genetic Algorithm (SSGA)

In Steady State Genetic Algorithm (SSGA), the current best chromosomes are automatically included in the next generation, and only the poorest chromosomes will be replaced. Therefore, SSGA allows some chromosomes to survive over time due to the Replacement process, because it allows some part of the current population to be carried to next generation, based on their fitness value (Mehra, et al., 2012). Figure (2.7) shows the SSGA structure:

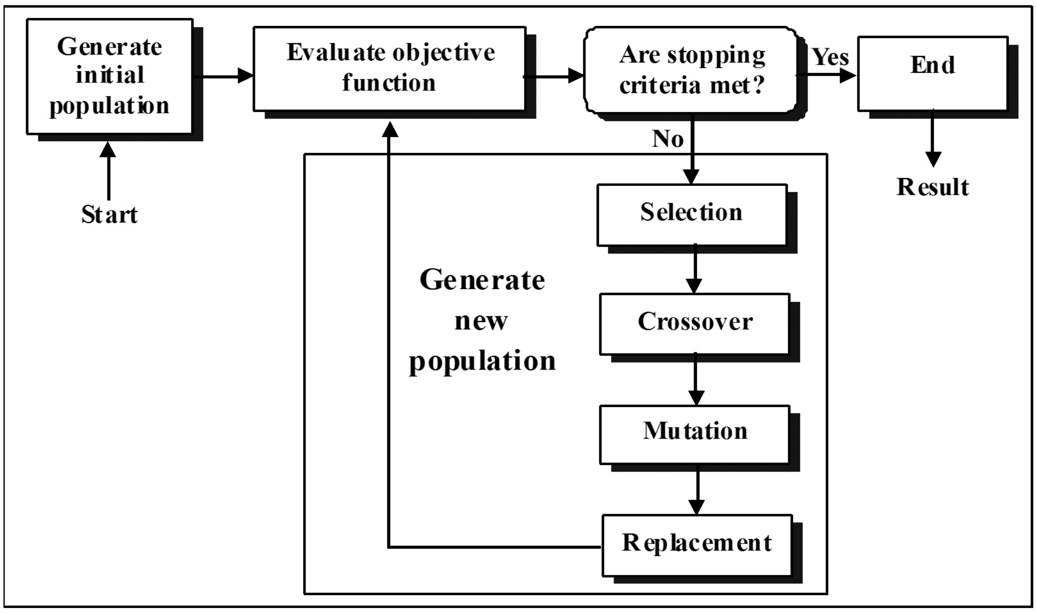

Generate initial population
Evaluate objective function
Are stopping criteria met?
Yes
End
Result
Start
No
Selection
Generate new population
Crossover
Mutation
Replacement

**Figure (2.7): Steady State Genetic Algorithm structure**

## 2.10 SGA versus SSGA

SGA and SSGA differ significantly in how individuals survive over time, how chromosomes are replaced, and how often they may reproduce. The replacement strategy likely to have a significant effect in producing advanced chromosomes due to the fact that it differs in SGA from SSGA.

There are many studies that were done in order to compare SGA and SSGA in variety fields. Duran and Xhafa, (2006) studied the effects of SGA and SSGA for Job Scheduling on Computational Grids in order to compare their behavior and study their practicability for a real grid application. The results show that SGA outperforms SSGA for the majority of instances, mainly for inconsistent and partially consistent matrices (which indicates that SGA performs better when job-machine constraints have to be managed). Both algorithms perform an accentuated reduction in time rapidly reaching good values, however SGA maintains more diversity among population thus reducing its tendency to converge and reaching better results than those of SSGA.

On other hand, Jones and Soule, (2006) compared Genetic robustness in SGA versus SSGA in “two peaks” problem; they concluded that although growth occurs with both algorithms, SSGA is able to converge on the higher peak without this growth. This result shows that the role of genetic robustness in the evolutionary process is significantly different in SGA versus SSGA.

This study will try to compare the power of SGA versus SSGA in intrusion detection field.

## 2.11 Network Intrusion Detection Dataset

In 1999, MIT Lincoln labs organized Knowledge Discovery in Databases cup (KDDcup99) and invited researchers across the world to design new techniques to build an IDS on training and testing data set which is referred to as the KDD cup 99 data set. KDD99 dataset has been the most widely used dataset for evaluation of computer network intrusion detection systems. It is built based on the data captured in DARPA’98 IDS evaluation program. The raw training data was about four gigabytes of compressed raw binary tcpdump data of seven weeks of network traffic, which can be processed into about five millions connection records, each with about 100 bytes. The two weeks of test data have around two millions connection records. Each record in KDD99 dataset contains 41 features and is labeled as either normal or attack, with exactly one specific attack type. The simulated attacks fall in one of the following four categories: DOS, U2R, R2L, and probing attacks (Hoque, et al, 2012) (Goyal and Kumar, 2008).

In 2009, Tavallaee, Bagheri, Lu, and Ghorbani, statistically analyzed the entire KDD data set. The analysis showed that there are some inherent problems in the KDD99 data set which highly affects the performance of evaluated systems. One of the most important deficiencies in the KDD data set is the huge number of redundant records. Analyzing KDD train and test sets found that about 78% and 75% of the records are duplicated in the train and test set, respectively. This large amount of redundant records in the train set will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning infrequent records which are usually more harmful to networks such as U2R attacks. The existence of these repeated records in the test set, on the other hand, will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records.

To solve these issues, Tavallaee, Bagheri, Lu and Ghorbani, (2009) proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD data set. This data set has advantages over the original KDD data set that it does not include redundant records in the training and testing datasets. The numbers of records in the train and test sets are 125,973 and 22,544 records, respectively. This research will use NSL-KDD dataset as environment to implement its experiments.

KDD'99 features can be classified into four groups: Basic features, Content features, Time-based Traffic Features, and Host-based Traffic Features (Kayacık, Zincir-Heywood, and Heywood. 2005).

1. **Basic features:** this category contains all the attributes that can be derived from a TCP/IP connection. Most of these features leading to an implicit delay in detection.
2. **Content features:** R2L and U2R attacks are embedded in the data portions of the packets, and normally involve only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.

3. **Time-based Traffic Features:** These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval.
4. **Host-based Traffic Features:** Utilize a historical window estimated over the number of connections - in this case 100 - instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

The label of the features and their corresponding network data features along with the categories are shown in Table (2.1). Each row represents a feature. The first column is the category of feature. Other columns show the feature name, feature description and type.

**Table (2.1): List of KDD99 Dataset features with their descriptions and data types (Amiri, Yousefi, Lucas, Shakery, and Yazdani, 2011)**

| # Category | Feature name | Description | Type |
|---|---|---|---|
| Category 1 | 1. duration | Length (number of seconds) of the connection | Continuous |
| | 2. protocol type | Type of the protocol, e.g., tcp, udp, etc. | Discrete |
| | 3. service | Network service on the destination, e.g., http, telnet, etc. | Discrete |
| | 4. flag | Normal or error status of the connection | Discrete |
| | 5. source bytes | Number of data bytes from source to destination | Continuous |
| | 6. destination bytes | Number of data bytes from destination to source | Continuous |
| | 7. land | 1 If connection is from/to the same host/ port; 0 otherwise | Discrete |
| | 8. wrong-fragment | Number of ‘‘wrong’’ fragments | Continuous |
| | 9. urgent | Number of urgent packets | Continuous |
| Category 2 | 10. hot | Number of ‘‘hot’’ indicators (hot: number of directory accesses, create and execute program) | Continuous |
| | 11. Num-failed-logins | Number of failed login attempts | Continuous |
| | 12. logged-in | 1 If successfully logged-in; 0 otherwise | Discrete |
| | 13. Num-compromised | Number of ‘‘compromised’’ conditions (compromised condition: number of file/path not found errors and jumping commands) | Continuous |
| | 14. root shell | 1 if root shell is obtained; 0 otherwise | Discrete |
| | 15. su-attempted | 1 if "su root" command attempted; 0 otherwise | Discrete |

| | | | |
|---|---|---|---|
| | 16. Num-root | Number of ‘‘root’’ accesses | Continuous |
| | 17. Num-file-creations | Number of file creation operations | Continuous |
| | 18. Num-shells | Number of shell prompts | Continuous |
| | 19. Num-access-files | Number of operations on access control files | Continuous |
| | 20. Num-outbound-cmds | Number of outbound commands in an ftp session | Continuous |
| | 21. Is-host-login | 1 If the login belongs to the ‘‘hot’’ list; 0 otherwise | Discrete |
| | 22. Is-guest-login | 1 If the login is a ‘‘guest’’login; 0 otherwise | Discrete |
| Category 3 | 23. Count | Number of connections to the same host as the current connection in the past 2 seconds | Continuous |
| | 24. Srv-count | Number of connections to the same service as the current connection in the past 2seconds (same-host connections) | Continuous |
| | 25. Serror-rate | % Of connections that have ‘‘SYN’’ errors (same-host connections) | Continuous |
| | 26. Srv-serror-rate | % Of connections that have ‘‘SYN’’ errors (same-service connections) | Continuous |
| | 27. Rerror-rate | % Of connections that have ‘‘REJ’’ errors (same-host connections) | Continuous |
| | 28. Srv-rerror-rate | % Of connections that have ‘‘REJ’’ errors (same-service connections) | Continuous |
| | 29. Same-srv-rate | % Of connections to the same service (same-host connections) | Continuous |
| | 30. Diff-srv-rate | % Of connections to different services (same-host connections) | Continuous |
| | 31. Srv-diff-host-rate | % Of connections to different hosts (same-service connections) | Continuous |
| Category 4 | 32. Dst-host-count | Count for destination host | Continuous |
| | 33. Dst-host-srv-count | Srv_count for destination host | Continuous |
| | 34. Dst-host-same-srv-rate | Same_srv_rate for destination host | Continuous |
| | 35. Dst-host-diff-srv-rate | Diff_srv_rate for destination host | Continuous |
| | 36. Dst-host-same-src-port-rate | Same_src_port_rate for destination host | Continuous |
| | 37. Dst-host-srv-diff-host-rate | Diff_host_rate for destination host | Continuous |
| | 38. Dst-host-serror-rate | Serror_rate for destination host | Continuous |
| | 39. Dst-host-srv-serror-rate | Srv_serror_rate for destination host | Continuous |
| | 40. Dst-host-rerror-rate | Rerror_rate for destination host | Continuous |
| | 41. Dst-host-srv-rerror-rate | Srv_serror_rate for destination host | Continuous |

## 2.12 Literature Review

This section discusses some of the techniques used for intrusion detection. The early effort in using GAs for IDS dates back to 1995, when Crosbie and Spafford implemented an IDS using multiple autonomous agents and Genetic Programming (GP) to detect network anomalies. Autonomous Agents are multiple functional entities that can work independently of each other and the system, each agent observes only one small aspect of the overall system. Genetic Programming is used as a learning paradigm to train those autonomous agents to detect potentially intrusive behaviors. This technique has the advantage of using many small autonomous agents, but the communication among them is still a problem and its training was time consuming if the agents are not appropriately initialized (Ojugo et al, 2012).

Zhao, Zhao and Li, (2005) used clustering genetic algorithms to build intelligent intrusion detection system. This system combines two stages into the process including clustering stage and genetic optimizing stage. In first stage, it used clustering analysis to build the initialized clustering sets by similarity rule. While in second stage, it used the genetic algorithm to optimize the clustering sets to distinguish the normal action and the intruded action. The empirical results showed that clustering genetic algorithm was successfully able to detect malicious intrusions in computer systems. This system got an overall accuracy level of 95%.

Zhou, Meng and Zhang, (2007) proposed an intrusion detection method based on GA and Support Vector Machine (SVM). They used GA to select and optimize features, then applied SVM model to classify and detect intrusions. The experimental results showed that SVM can achieve good classification accuracy, and the accuracy can be improved after feature selection and optimization. Therefore, it is efficient to apply SVM and GA to intrusion detection. KDD99 dataset is used to train and test this approach.

Al-Sharafat and Naoum, (2009) incorporated different techniques into a classifier system to detect and classify intrusion from normal network connection. They used SSGA as a discovery mechanism and Zeroth Level Classifier System as detector by matching incoming connection with classifiers to determine whether the current connection is normal or intrusion, while Fuzzy Logic is used to optimize crossover and mutation probability. KDD99 dataset is used to perform the experiments and evaluations of the proposed method.

Shirazi and Kalaji, (2010) implemented information theory measures in order to select the most significant features by ranking the network connection features according to their importance in detecting attacks. This ranking allows reducing the computing complexity and decreasing the detection speed without affecting the detection rates by selecting the most significant features for each attack class. Then, the authors designed the network traffic linear classifiers based on genetic algorithms by using the top five features according to their importance in detecting attack.

Hoque, Mukit and Bikas (2012) presented and implemented a misuse based NIDS using SGA to efficiently detect various types of network intrusions. This approach used evolution theory to information evolution in order to filter the traffic data and thus reduce the complexity and also used the standard KDD99 benchmark dataset to implement and measure the performance of their system. The authors used only the numerical features, both continuous and discrete, also used the standard deviation equation with distance to measure the fitness of a chromosome. They got the following Detection Rate results (Probe: 71.1%, DoS: 99.4%, U2R: 18.9% and R2L: 5.4%).

Goyal, Aggarwal and Jain, (2012) showed effect of change in rate of genetic algorithm operators on fitness value in production of rules for misuse ID. In this approach, six features were taken to compose a classification rules, they were source IP, destination IP, destination port, protocol, sender data amount, and responder data amount. The support and confidence framework is used as fitness function. A crossover rates applied in this study are 0%, 30%, and 80%. This study concluded that if crossover rate is not sufficient, there is no sufficient exchange of genes. If crossover rate is too much, good parts of individuals get split up a lot. This allows some individuals with high fitness values to be copied directly to the next population. Also a lot of mutation would break up good genes and stop them from being passed on.

Alabsi, (2012) proposed new fitness function, Reward Penality Fitness Function, to be used in the evaluation process. He also compared selection and crossover methods to choose the best one to implement it in a system; and found that Uniform Crossover is the best one among Crossover types and it is better to be combined with Stochastic Universal Sampling Selection (SUS) or Elitist Selection methods. This study used SUS Selection and Uniform Crossover as parameters in SSGA to be implemented in misuse NIDS. The proposed system got an average of DR equal to 95% and an average of FPR equal to 0.297%.

Jongsuebsuk, Wattanapongsakorn, and Charnsripinyo (2013) developed a real-time detection approach for detecting anomaly attacks. They used packet sniffer to sniff network packets in every 2 seconds and preprocessed it into 12 features and used Fuzzy Genetic algorithm to classify the network data. The fuzzy rule is a supervised learning technique and genetic algorithm make fuzzy rule able to learn new attacks by itself. The output can be categorized into DoS and Probe. They used network dataset for training and testing is collected in the actual network environment in their research laboratory. The result shows that this algorithm has over 97% of DR, less than 1% of FPR and less than 3 seconds (for data preprocessing and detection) to issue the alert message after an attack arrival.

Torkaman, Javadzadeh, and Bahrololum, (2013) designed a hybrid approach for modeling HIDS combines anomaly and misuse detection, based on two-layer fuzzy Genetic algorithm and neural network which uses simple data mining techniques to process the web application traffics, Two-layer fuzzy Genetic algorithm and neural network are applied respectively as anomaly and misuse detection. One of the advantages of this algorithm is that it can support multiple attack classification according to Open Web Application Security Project (OWASP). This research used the HTTP dataset CSIC 2010 which is generated automatically and contains 36,000 normal requests and more than 25,000 anomalous requests. The proposed model is able to detect critical vulnerabilities based on OWASP standard.

Aziz, (2014) enhanced SSGA based IDS for detecting misuse attacks by comparing Replacement method. He declared that Triple Tournament Replacement (TTR) produces more accurate results than Binary Tournament Replacement (BTR) according to the value of DR and the number of new rules. Also, he found that TTR enhanced the convergence to the solution and improved the efficiency of SSGA for producing new rules. This research got an average of DR that is equal to 88.25%, and an average of FPR that is equal to 1.48%. The experiments and evaluations are performed by using 10% of the KDD99 dataset. But this study will enhance IDS by comparing SGA and SSGA in intrusion detection field.

Pal and Parashar, (2014) proposed IDS using modified GA , they applied attribute subset reduction on the basis of Information gain in order to reduce the complexity and training time. Then, Normalize and Fuzzify each selected attribute and divide into fuzzy classes is done by using the triangular function. This research concluded that Fuzzy-genetic Intrusion detection system combined with feature selection enable the system to produce optimal subset of attribute in the midst of huge network information. Also, embedding a soft computing approach in rule generation makes the rule more efficient than hard computing. This approach was verified using KDD'99 intrusion detection data set. Experimental result showed that the proposed method achieved high DR and low FPR.

Ghosh and Mitra, (2015) proposed an efficient IDS by applying the concept of GA with Best Feature Set Selection (BFSS) method for feature selection and Logistic Regression (LR) for classification to detect network intrusions. The authors used GA to select a number of feature sets, where each chromosome represents a particular set of features, then their proposed BFSS method is applied to select the best set of those feature sets obtained from GA results. After selecting the most relevant features from NSL-KDD data set, they built a classifier using LR with Gradient Descent as an optimization algorithm to detect both anomaly and misuse attacks.

# Chapter Three

# Methodology & Proposed Models

## 3.1 Methodology

This research aims at verifying the power of Simple Genetic Algorithm (SGA) against Steady State Genetic Algorithm (SSGA) for Intrusion Detection System (IDS) by measuring the performance of each model in detection both abnormal and normal behavior and if abnormal is detected, find which type of attack it is.

The NSL-KDD dataset will be used as an environment to train and test the proposed models, on which the system will use the entire training dataset and the entire testing dataset from the NSL-KDD dataset.

This research has two proposed models. In the first proposed model, the researcher will use SGA to develop IDS. In the second proposed model, the researcher will use SSGA instead of SGA to develop IDS.

The proposed design for both models, SGA based IDS and SSGA based IDS, consists of three phases: Processing phase, Genetic Algorithm phase, and Testing phase, as shown in figure (3.1) and figure (3.2). The following sections discuss those phases in details.

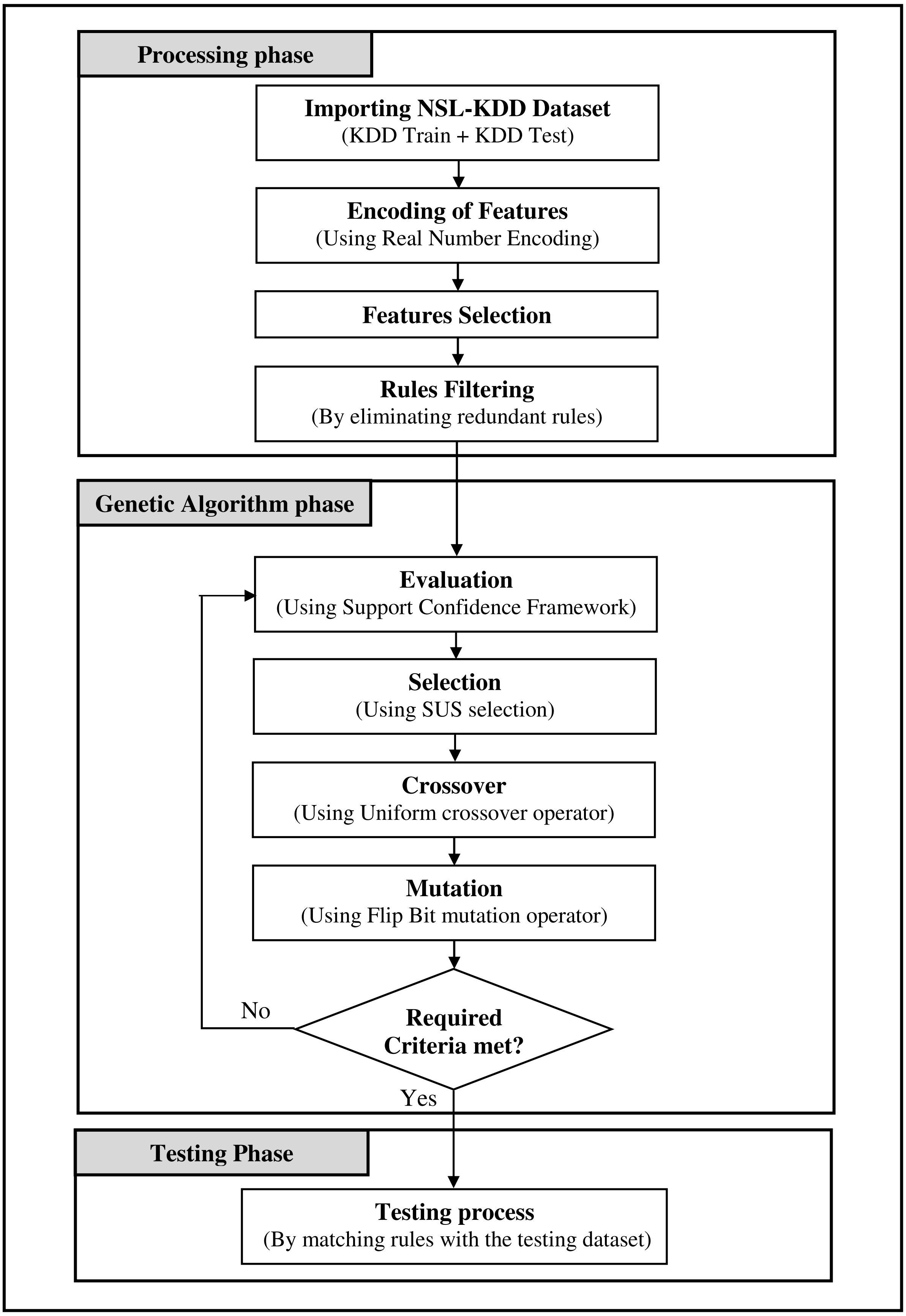

Processing phase
Importing NSL-KDD Dataset
(KDD Train + KDD Test)
Encoding of Features
(Using Real Number Encoding)
Features Selection
Rules Filtering
(By eliminating redundant rules)
Genetic Algorithm phase
Evaluation
(Using Support Confidence Framework)
Selection
(Using SUS selection)
Crossover
(Using Uniform crossover operator)
Mutation
(Using Flip Bit mutation operator)
Required
Criteria met?
No
Yes
Testing Phase
Testing process
(By matching rules with the testing dataset)

**Figure (3.1): The structure of SGA based IDS**

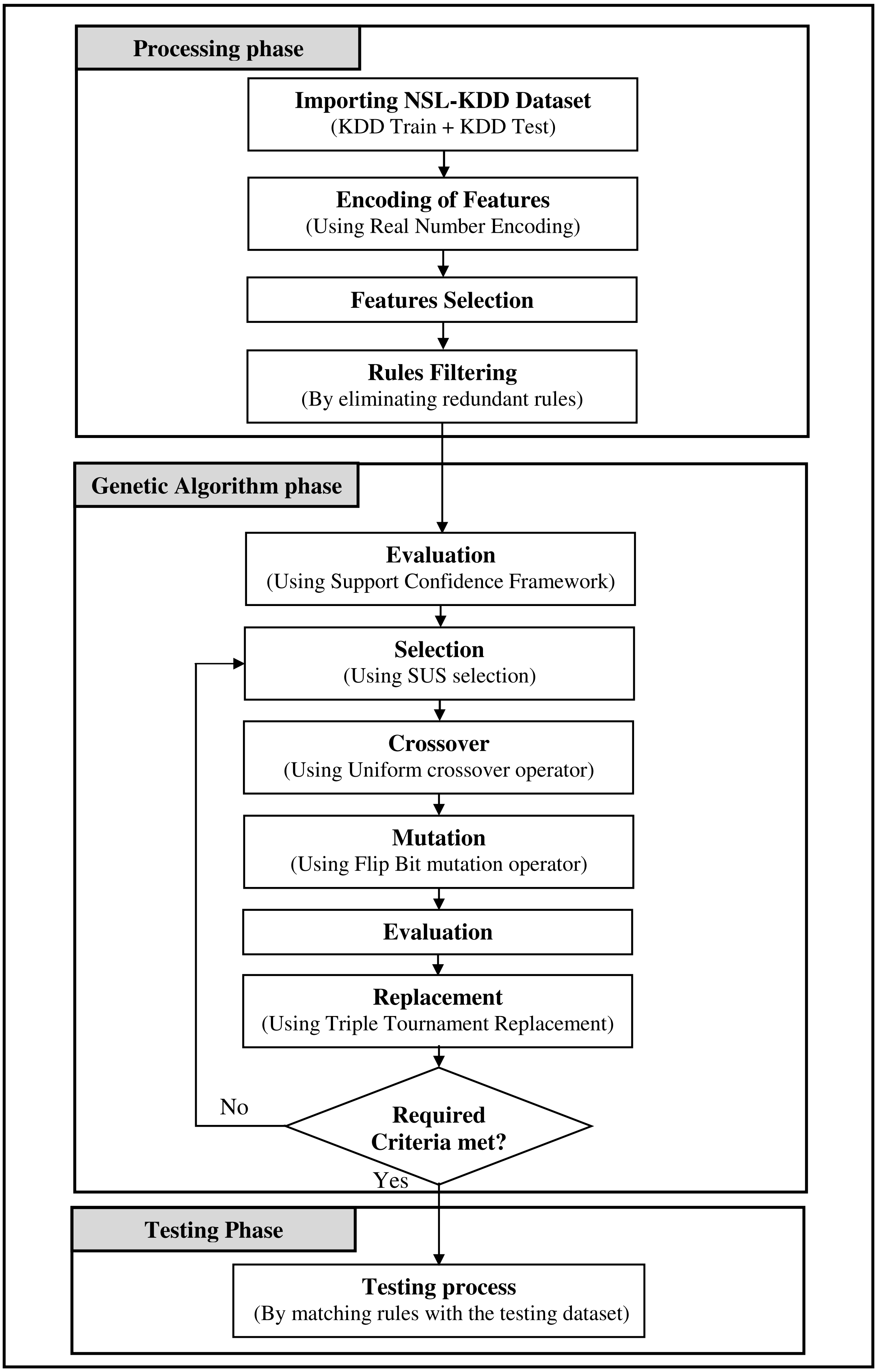

Processing phase
Importing NSL-KDD Dataset
(KDD Train + KDD Test)
Encoding of Features
(Using Real Number Encoding)
Features Selection
Rules Filtering
(By eliminating redundant rules)
Genetic Algorithm phase
Evaluation
(Using Support Confidence Framework)
Selection
(Using SUS selection)
Crossover
(Using Uniform crossover operator)
Mutation
(Using Flip Bit mutation operator)
Evaluation
Replacement
(Using Triple Tournament Replacement)
Required Criteria met?
No
Yes
Testing Phase
Testing process
(By matching rules with the testing dataset)

**Figure (3.2): The structure of SSGA based IDS**

## 3.2 Processing Phase

This phase starts with importing the dataset and the encoding of features, selecting the dataset with the reduced features for each attack, then filtering the duplicated rules by eliminating redundant ones.

### 3.2.1 NSL-KDD dataset

The NSL-KDD dataset will be used as an environment to train and test the proposed models. It consists of two datasets: training dataset and testing dataset.

The training dataset will be used to build the rules that are used to detect attack connections. This research will use the entire NSL-KDD training dataset, which consists of 125,973 records.

The testing dataset will be used to evaluate the proposed models efficiency by measuring DR and FPR for each model. This research will use the entire NSL-KDD testing dataset, which consists of 22,544 records.

### 3.2.2 Encoding of Features

Every network connection in NSL-KDD dataset contains 41 features, three of which carry  string data, 15 features carry float type values in the range 0.00-1.00, while the remaining 23 features carry integer values. In this research, real number encoding will be used to represent rules because the structure of genotype in real number encoding is identical to that of the phenotype as well as real number encoding already consists of float numbers and integer numbers. Therefore, only features with string data are needed to be represented in real number form.

### 3.2.3 Features Selection

Features selection is a key identification feature method for building efficient and effective IDS by filtering out noise and removing redundant and irrelevant features. Each record in NSL-KDD dataset is described by 41 features, using all these features to generate rules is a time consuming process. So, the most significant features should be selected to represent each attack category.

There are several studies that have proposed different features sets to represent different type of attacks. Table (3.1) shows four of those researches and the selected features for each attack type.

**Table (3.1): The most suitable features for each attack**

| Research | DoS | Probe | U2R | R2L |
|---|---|---|---|---|
| Mukkamala, et al.,(2004) | 7, 8, 12, 13, 23 | 3, 12, 27, 31, 35 | 14, 17, 25, 36, 38 | 6, 11, 12, 19, 22 |
| Chou, et al., (2008) | 1, 2, 3, 4, 5, 6, 12, 23, 24, 31, 32, 37 | 1, 2, 3, 4, 12, 16, 25, 27, 28, 29, 30, 40 | 1, 2, 3, 10, 16 | 1, 2, 3, 4, 5, 10, 22 |
| Amiri, et al., (2011) Using FFSA | 5, 38, 3 | 40, 5, 41, 11, 2, 22, 9, 27, 37, 28, 14, 19, 31, 18, 1, 17, 16, 13, 25, 39, 26, 6, 30, 32 | 5, 1, 19, 18, 39, 2, 22, 9, 29, 7, 8, 15, 30, 16, 20, 21, 6, 3, 26, 31, 33, 14, 4, 17, 32, 12, 25 | 3, 6, 4, 11, 9, 33, 37, 38, 22, 25 |
| Amiri, et al., (2011) Using MMIFS | 5, 23, 6, 2, 24, 41, 36, 3 | 40, 5, 33, 23, 28, 3, 41, 35, 27, 32, 12, 24 | 5, 1, 3, 24, 23, 2, 33, 6, 32, 4, 14, 21 | 3, 13, 22, 23, 10, 5, 35, 24, 6, 33, 37, 32, 1, 39 |

In order to select the most significant features, the researcher performed the testing process over the testing dataset using the features sets shown in table (3.1), and selected the best group among these features. More details will be discussed further in chapter four.

### 3.2.4 Rules Filtering

After selecting the training dataset with the reduced features for each attack, duplicated rules might appear. These duplicated rules are unnecessary and keeping them could slow the work down. So, training records will be filtered by eliminating the redundant rules.

After analyzing the training dataset, each rule will be represented in: *if* condition *then* action. The condition part refers to the features of the network connection, the result might be TRUE if the incoming connection matched the rules in dataset, or it could be FALSE if there was some mismatching. While the action part refers to the attack name and will be specified only if condition is true. For example, a rule can be defined as:

***If*** (protocol type =" tcp") and (Number of data bytes from source to destination = 0) and (Percentage of connections to the current host and specified service that have an RST error = 0.99) ***then*** (Current connection = "Neptune attack")

## 3.3 Genetic Algorithm Phase

The main difference between the two models proposed in this study is Genetic Algorithm phase, especially in Replacement process. In this phase, SGA based IDS model will perform Evaluation, Selection, Crossover and Mutation processes. While

SSGA based IDS model will perform Evaluation, Selection, Crossover, Mutation, as well as Replacement processes.

In this phase, GA processes will be applied on the filtered dataset in order to generate new rules which will be used to identify attacks from the testing dataset.

### 3.3.1 Evaluation

Support Confidence Framework will be adopted as the fitness function in this research in order to evaluate each individual in the training dataset. This fitness function was developed by Wong and Leung (2000). Positive results were obtained when the Support Confidence Framework was used by many researches such as Lu and Traore (2004), Selvakani and Rajesh (2007), and Ojugo, et al., (2012).

To explain the work of Support Confidence fitness function, suppose that a rule is represented as (*if* condition *then* action), where a condition part represents the features values while an action part represents the attack name. if the condition and action of the selected record equal to the condition and action of the compared record in training dataset, then this will increase the value of (A and B) of the selected record by one. Else, if the condition part of the selected record is equal to the condition part of the compared record but the actions of both records don't meet each other, then the value of (A) of the selected record will increase by one. To calculate a fitness value for each rule in the training dataset using Support Confidence fitness function, must:

1. Calculate the Support value by computing the rate of the (A and B) value to the number of records in training dataset (N). Support value is calculated by using equation (3.1):

$$\text{Support} = |\text{A and B}| / N \qquad (3.1)$$

Where:

|A and B| = Number of records matching the condition part and action part.

N = Number of connections in training dataset.

2. Calculate the Confidence value by computing the rate of the (A and B) value to the (A) value. Confidence value is calculated by using equation (3.2):

$$\text{Confidence} = |A \text{ and } B| / |A| \qquad (3.2)$$

Where:

|A| = Number of records matching the condition part only.

3. Calculate the fitness value by computing Support Confidence fitness function, it is calculated by using equation (3.3):

$$\text{Fitness} = w_1 \times \text{Support} + w_2 \times \text{Confidence} \qquad (3.3)$$

Where:

$w_1$, $w_2$ = Weights to balance the two terms.

### 3.3.2 Selection

Stochastic Universal Sampling (SUS) will be used as the selection method. SUS was developed by Baker and it became one of the most widely used selection methods because it has zero bias (Pencheva, Atanassov, and Shannon, 2009).

### 3.3.3 Crossover

Uniform crossover operator will be used in the research; the uniform crossover operator is considered the most powerful crossover because all genes have equal probability to be swapped in order to gain a high diversity in population (Hu and Di Paolo, 2009).

### 3.3.4 Mutation

**F**lip Bit mutation will also be used in this research. In Flip Bit mutation a gene is randomly selected and its value is inverted to be equal to a random number of a specific range.

### 3.3.5 Replacement

This step will be used only with SSGA based IDS model, the Triple Tournament Replacement (TTR) will be adopted in this study because it produces more accurate results than the double replacement (Aziz, 2014). In this method, three individuals are chosen to compete and the individual with the highest fitness will win.

### 3.3.6 Stopping Criteria

GA is iterative process and will be stopped when the stopping criteria is achieved. The stopping criteria used in this research is if there are no additional new rules to be produced, then GA will be stopped.

## 3.4 Testing Phase

After applying Genetic Algorithm process over the training dataset and generating new rules. These generated rules will be used to detect attacks from the testing dataset. By calculating DR and FPR for each proposed model, the best model will be identified.

## 3.5 Proposed Models Evaluation

An effective IDS must be able to detect various types of intrusions with high DR and low FPR. Testing proposed methods can provide a good indicator on whether these methods can give high performance compared with others or not. Evaluate IDS can be expressed as how far it can correctly detect intrusions and avoid false alarm.

The two models proposed in this study will be evaluated by calculating Detection Rate (DR) and False Positive Rate (FPR). Hoque, et al, (2012) defined DR and FPR as following:

**Detection Rate (DR)** is "the ratio between the number of correctly detected intrusions and the total number of intrusions". The DR is calculated by using equation (3.4).

$$\mathrm{DR} = \frac{\text{No. of Detected Attacks}}{\text{No. of Total Attacks}} \qquad (3.4)$$

**False Positive Rate (FPR)** is "the ratio between the numbers of normal connections that are incorrectly classifies as intrusions and the total number of normal connections". The FPR is computed by using equation (3.5).

$$\mathrm{FPR} = \frac{\text{No. of False Alarms}}{\text{No. of Normal Records}} \qquad (3.5)$$

# Chapter Four

# Systems Structure & Experimental Results

## 4.1 Overview

This chapter presents the implementation stages of the proposed models, starting with analyzing the dataset, selecting rules with reduced features for each attack type and then eliminating duplicated rules. This chapter also presents the applying of SGA and SSGA on the training dataset in order to generate new rules which help to detect attacks in testing dataset. Finally, presenting experimental results and providing a concise discussion for those results.

The researcher used Microsoft Visual Basic 2010 Express as the front end to write the coding part, and also used Microsoft SQL Server 2008 as the back end to the system, in order to store and analyze the dataset. For this implementation, the researcher used Windows based HP computer that has an i5 core (1.7 GHz) processor, 500 GB hard disk space and 4 GB RAM.

As mention early in chapter three, this study proposed two models: SGA based IDS and SSGA based IDS. Each model consists of three phases: Processing phase, Genetic Algorithm phase, and Testing phase. Section (4.2), (4.3), and (4.4) will present the implementation stages of those phases step by step.

## 4.2 Processing Phase

In this phase, the dataset is imported and encoded using Real Number Encoding, each attack rules is selected with reduced features and the duplicated rules are filtered by eliminating redundant rules. Figure (4.1) shows the processes of this phase.

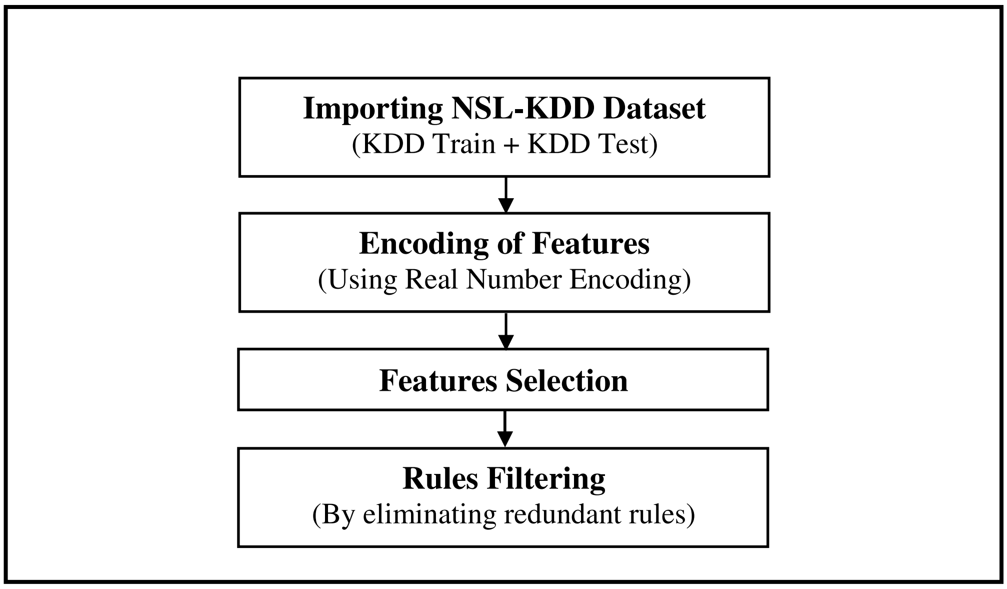
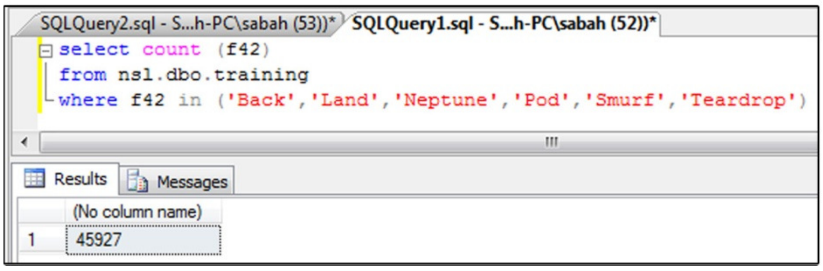

Importing NSL-KDD Dataset
(KDD Train + KDD Test)
Encoding of Features
(Using Real Number Encoding)
Features Selection
Rules Filtering
(By eliminating redundant rules)

**Figure (4.1): processing phase major compenents**

### 4.2.1 NSL-KDD Dataset

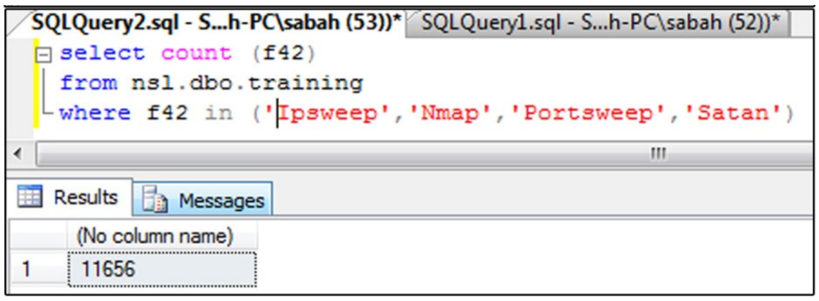
NSL-KDD dataset is a benchmark used to evaluate the system efficiency by measuring its performance; it is publicly available on (http://nsl.cs.unb.ca/NSL-KDD/). It consists of training dataset and testing dataset. The training dataset contains 125,973 records, while the testing dataset contains 22,544 records. Figure (4.2) shows the analysis of NSL-KDD dataset and its results of training and testing datasets, on table (4.1) and table (4.2), respectively.


SQLQuery2.sql - S...h-PC\sabah (53))*
SQLQuery1.sql - S...h-PC\sabah (52))*
Results
Messages

```
select count (f42)
from nsl.dbo.training
where f42 in ('Back','Land','Neptune','Pod','Smurf','Teardrop')
```

```
  (No column name)
1 45927
```

a: analyzing DoS attack types


SQLQuery2.sql - S...h-PC\sabah (53))*
SQLQuery1.sql - S...h-PC\sabah (52))*
Results
Messages

```
select count (f42)
from nsl.dbo.training
where f42 in ('Ipsweep','Nmap','Portsweep','Satan')
```

```
  (No column name)
1 11656
```

b: analyzing Probe attack types

**Figure (4.2): analyzing NSL-KDD dataset using Microsoft SQL Server 2008**

**Table (4.1): details information about training dataset**

| Attack name | Sub attack name | No. of records |
|---|---|---|
| Normal | Normal | 67,343 |
| Dos | Back , Land , Neptune , Pod , Smurf , Teardrop | 45,927 |
| Probe | Ipsweep , Nmap , Portsweep , Satan | 11,656 |
| U2R | buffer_overflow , Loadmodule , Perl , Rootkit | 52 |
| R2L | ftp_write , guess_passwd , Imap , Multihop , Phf , Spy ,Warezmaster , Warezclient | 995 |
| Total | | 125,973 |

**Table (4.2): details information about testing dataset**

| Attack name | Sub attack name | No. of records |
|---|---|---|
| Normal | Normal | 9711 |
| Dos | Back , Land , Neptune , Pod , Smurf , Teardrop, apache2 , Udpstorm , Mailbomb , Processtable | 7458 |
| Probe | Ipsweep , Nmap , Portsweep , Satan , Saint , Mscan | 2421 |
| U2R | buffer_overflow , Loadmodule , Perl , Rootkit , Ps , Xterm , Sqlattack | 67 |
| R2L | ftp_write , guess_passwd , Imap , Multihop , Phf , Warezmaster , Named , Sendmail , Xlock , Xsnoop , Snmpgetattack , Httptunnel , Worm , Snmpguess | 2887 |
| Total | | 22,544 |

As shown in the above tables, Warezclient and Spy attacks are existent only in training dataset. However, on the contrary there are some attacks that exist only in testing dataset, such as apache2, udpstorm, mailbomb, processtable…etc. thus; such attacks were not used in this research.

## 4.2.2 Encoding of Features

In NSL-KD dataset each rule contains 41 features, three of which are of String values that would be encoded using Real Number Encoding, those features are Protocol Type, Flag and Service. Table (4.3) represents those features as follows:

- **Protocol type feature** has three different values; TCP, ICMP and UDP. So, TCP is encoded by number 1, ICMP is encoded by number 2 and UDP is encoded by number 3.
- **Flag feature** has 11 different values; each value was represented by a positive number in range of [1-11]. "REJ" value which was represented by number 1, "OTH" by number 2, "S1" by number 3, "RSTR" by number 4, "S2" by number 5, "S3" by number 6, "S0" by number 7, "SF" by number 8, "SH" by number 9, "RSTO" by number 10, and finally "RSTOS0" was represented by number 11.
- **Service feature** has 70 different values. The researcher used positive numbers to represent these values in range of [1-70], such as "BGP" value which was represented by number 1, "SQL_NET" by number 2 … etc.

Figures (4.3) and (4.4) present the network connection features in NSL-KDD dataset before and after encoding, respectively.

1, **tcp**, **smtp**, **SF**, 1710, 375, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 184, 128, 0.35, 0.39, 0.01, 0.02, 0, 0, 0.32, 0

**Figure (4.3): the network connection features before encoding**

1, **1**, **60**, **8**, 1710, 375, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 184, 128, 0.35, 0.39, 0.01, 0.02, 0, 0, 0.32, 0

**Figure (4.3): the network connection features after encoding**

**Table (4.3): the encoding of the string features of NSL-KDD dataset**

**Protocol type Feature**

| String value | Encoding |
|---|---|
| Tcp | 1 |
| Icmp | 2 |
| Udp | 3 |

**Flag Feature**

| String value | Encoding |
|---|---|
| REJ | 1 |
| OTH | 2 |
| S1 | 3 |
| RSTR | 4 |
| S2 | 5 |
| S3 | 6 |
| S0 | 7 |
| SF | 8 |
| SH | 9 |
| RSTO | 10 |
| RSTOS0 | 11 |

**Service Feature**

| String value | Encoding | String value | Encoding | String value | Encoding |
|---|---|---|---|---|---|
| Bgp | 1 | Systat | 25 | Domain | 49 |
| sql_net | 2 | Time | 26 | Kshell | 50 |
| Nnsp | 3 | Gopher | 27 | http_443 | 51 |
| Mtp | 4 | telnet | 28 | Auth | 52 |
| remote_job | 5 | Netstat | 29 | Echo | 53 |
| ecr_i | 6 | Whois | 30 | Name | 54 |
| Finger | 7 | http_8001 | 31 | urp_i | 55 |
| Uucp | 8 | Efs | 32 | tftp_u | 56 |
| X11 | 9 | Aol | 33 | netbios_ssn | 57 |
| Link | 10 | Ssh | 34 | eco_i | 58 |
| Z39_50 | 11 | Supdup | 35 | iso_tsap | 59 |
| ntp_u | 12 | Vmnet | 36 | Smtp | 60 |
| Hostnames | 13 | http | 37 | Harvest | 61 |
| netbios_dgm | 14 | urh_i | 38 | domain_u | 62 |
| IRC | 15 | Daytime | 39 | Other | 63 |
| Rje | 16 | pop_2 | 40 | Discard | 64 |
| tim_i | 17 | Exec | 41 | nntp | 65 |
| red_i | 18 | imap4 | 42 | Ldap | 66 |
| pm_dump | 19 | Printer | 43 | ftp | 67 |
| Klogin | 20 | Private | 44 | Ctf | 68 |
| Login | 21 | csnet_ns | 45 | Sunrpc | 69 |
| http_2784 | 22 | pop_3 | 46 | Courier | 70 |
| Shell | 23 | uucp_path | 47 | | |
| ftp_data | 24 | netbios_ns | 48 | | |

### 4.2.3 Features Selection

In order to select the most significant features, the researcher used the features sets shown in table (3.1) to test the testing dataset. The features sets that achieve high DR and low FPR than other sets will be selected to use them in this research.

Table (4.4) shows the results of testing the selected features by Mukkamala, at al., (2004) to identify attacks from testing dataset.

**Table (4.4): results of DR and FPR for each attack using the selected features by (Mukkamala, at al., 2004)**

| | Dos | Probe | U2R | R2L |
|---|---|---|---|---|
| The selected features | 7, 8, 12, 13, 23 | 3, 12, 27, 31, 35 | 14, 17, 25, 36, 38 | 6, 11, 12, 19, 22 |
| DR | 93.68% | 44.65% | 19.40% | 15.66% |
| FPR | 67.97% | 7.24% | 38.56% | 7.68% |

Table (4.5) shows the results of testing the selected features by Chou, et al., (2008) to identify attacks from testing dataset.

**Table (4.5): results of DR and FPR for each attack using the selected features by (Chou, et al., 2008)**

| | Dos | Probe | U2R | R2L |
|---|---|---|---|---|
| The selected features | 1, 2, 3, 4, 5, 6, 12, 23, 24, 31, 32, 37 | 1, 2, 3, 4, 12, 16, 25, 27, 28, 29, 30, 40 | 1, 2, 3, 10, 16 | 1, 2, 3, 4, 5, 10, 22 |
| DR | 30.77% | 30.24% | 10.45% | 0.10% |
| FPR | 0.35% | 10.98% | 3.45% | 0.00% |

Amiri, et al., (2011) proposed two features sets using Modified Mutual Information Feature Selection algorithm (MMIFS) and Forward Feature Selection Algorithm (FFSA), as shown in table (3.1) in chapter three. The researcher used these two feature sets to identify attacks from testing dataset and got the results shown in table (4.6) and (4.7).

**Table (4.6): results of DR and FPR for each attack according to (Amiri, et al., 2011) using the selected features by FFSA**

| | Dos | Probe | U2R | R2L |
|---|---|---|---|---|
| The Selected features | 5, 38, 3 | 40, 5, 41, 11, 2, 22, 9, 27, 37, 28, 14, 19, 31, 18, 1, 17, 16, 13, 25, 39, 26, 6, 30, 32 | 5, 1, 19, 18, 39, 2, 22, 9, 29, 7, 8, 15, 30, 16, 20, 21, 6, 3, 26, 31, 33, 14, 4, 17, 32, 12, 25 | 3, 6, 4, 11, 9, 33, 37, 38, 22, 25 |
| DR | 73.38% | 27.67% | 0.00% | 13.61% |
| FPR | 1.82% | 0.71% | 0.00% | 1.02% |

**Table (4.7): results of DR and FPR for each attack according to (Amiri, et al., 2011) using the selected features by MMIFS**

| | Dos | Probe | U2R | R2L |
|---|---|---|---|---|
| The Selected features | 5, 23, 6, 2, 24, 41, 36, 3 | 40, 5, 33, 23, 28, 3, 41, 35, 27, 32, 12, 24, | 5, 1, 3, 24, 23, 2, 33, 6, 32, 4, 14, 21 | 3, 13, 22, 23, 10, 5, 35, 24, 6, 33, 32, 1, 37, 39 |
| DR | 28.87% | 6.65% | 0.00% | 0.00% |
| FPR | 0.35% | 0.01% | 0.00% | 0.00% |

According to the results shown in the previous tables and taking into account the ratios of DR and FPR, where the good feature set must to achieve high DR and low FPR. The researcher selected the features sets shown in table (4.8) to use them in his research.

**Table (4.8): the selected features sets for this research**

| Attack name | Features sets |
|---|---|
| Dos | F3, F5, F38 |
| Probe | F3, F12, F27, F31, F35 |
| U2R | F1, F2, F3, F10, F16 |
| R2l | F3, F4, F6, F9,F11, F22, F25, F33, F37, F38 |

## 4.2.4 Rules Filtering

After selecting each sub attack rules with reduced features, many duplicated rules have appeared, those rules were filtered by eliminating the redundant ones. Table (4.9) and table (4.10) show the number of rules before and after filtering in DoS attacks category and Probe attacks category, respectively. While table (4.11) and table (4.12) show the number of rules before and after filtering for U2R and R2L attacks, respectively.

**Table (4.9): number of rules before and after filtering for DoS attacks category**

| Sub attack name | No. of rules before filtering | No. of rules after filtering |
|---|---|---|
| Back | 956 | 50 |
| Land | 18 | 5 |
| Pod | 201 | 15 |
| Smurf | 2646 | 43 |
| Teardrop | 892 | 5 |
| Neptune | 41214 | 226 |

**Table (4.10): number of rules before and after filtering for Probe attacks category**

| Sub attack name | No. of rules before filtering | No. of rules after filtering |
|---|---|---|
| Ipsweep | 3599 | 40 |
| Nmap | 1493 | 97 |
| Portsweep | 2931 | 547 |
| Satan | 3633 | 671 |

**Table (4.11): number of rules before and after filtering for U2R attacks category**

| Sub attack name | No. of rules before filtering | No. of rules after filtering |
|---|---|---|
| buffer_overflow | 30 | 24 |
| Loadmodule | 9 | 8 |
| Perl | 3 | 3 |
| Rootkit | 10 | 7 |

**Table (4.12): number of rules before and after filtering for R2L attacks category**

| Sub attack name | No. of rules before filtering | No. of rules after filtering |
|---|---|---|
| ftp_write | 8 | 8 |
| guess_passwd | 53 | 53 |
| Imap | 11 | 11 |
| Multihop | 7 | 7 |
| Phf | 4 | 4 |
| Warezmaster | 20 | 20 |

## 4.3 Genetic Algorithm Phase

This phase aims at generating new rules to be used for detecting attacks in testing dataset. In this phase, SGA based IDS applied Evaluation, Selection, Crossover and Mutation processes, while SSGA Based IDS performed Evaluation, Selection, Crossover, Mutation as well as Replacement processes.

### 4.3.1 Evaluation

The fitness function adopted in the research is called Support Confidence Framework. To demonstrate the way it operates, consider the following example:

Suppose that table (4.13) represents all records of the training dataset, to calculate the fitness value of record1, must to compute values of N, |A|, and |A and B|. Where:

N = Number of connections in training dataset.

|A| = Number of records matching the condition part only.

|A and B| = Number of records matching the condition part and action part.

Therefore:

N= 10

|A| = 4 (record 1, record 3, record 6 and record 9).

|A and B| = 2 (record1, record 3).

Depending on the equations (3.1), (3.2) and (3.3) in chapter three, the fitness value for record1 is:

Support = |A and B| / N =2/10 = 0.2

Confidence = |A and B| / |A| = 2/4 = 0.5

Fitness value = $w_1 \times$ Support + $w_2 \times$ Confidence = $(0.2 \times 0.2) + (0.8 \times 0.5) = 0.44$

**Table (4.13): working of Support Confidence Framework**

| ID | F1 | F2 | F3 | F10 | F16 | Attack name |
|---|---|---|---|---|---|---|
| **1** | **25** | **1** | **28** | **0** | **2** | **Perl** |
| 2 | 60 | 1 | 28 | 0 | 0 | Rootkit |
| **3** | **25** | **1** | **28** | **0** | **2** | **Perl** |
| 4 | 7 | 1 | 67 | 4 | 0 | Loadmodule |
| 5 | 290 | 1 | 28 | 3 | 4 | Buffer_overflow |
| **6** | **25** | **1** | **28** | **0** | **2** | Rootkit |
| 7 | 98 | 1 | 28 | 1 | 14 | Rootkit |
| 8 | 0 | 1 | 24 | 1 | 0 | Loadmodule |
| **9** | **25** | **1** | **28** | **0** | **2** | Buffer_overflow |
| 10 | 7 | 1 | 67 | 4 | 0 | Perl |

Table (4.14) shows the actual data of fitness values for Rootkit attack rules using Support Confidence Framework, where the values used for w1 and w2 in this research were 0.2 and 0.8, respectively. For example, the fitness value of first record in table (4.14) is calculated as:

N= 125,973, |A| = 1, |A and B| = 1, $w_1 = 0.2$, $w_2 = 0.8$

Support = |A and B| / N =1/125,973 = 0.0000079

Confidence = |A and B| / |A| = 1/1 = 1

Fitness value = $w_1$ × Support + $w_2$ × Confidence

= (0.2 × 0.0000079) + (0.8 × 1) = 0.8000016

**Table (4.14): Rootkit attack rules with reduced features and fitness values (actual data)**

| ID | F1 | F2 | F3 | F10 | F16 | Fitness value |
|---|---|---|---|---|---|---|
| 1 | 708 | 1 | 28 | 0 | 7 | 0.8000016 |
| 2 | 98 | 1 | 28 | 1 | 14 | 0.8000016 |
| 3 | 0 | 3 | 63 | 0 | 0 | 0.002373963 |
| 4 | 60 | 1 | 28 | 0 | 0 | 0.3200032 |
| 5 | 60 | 1 | 28 | 0 | 0 | 0.3200032 |
| 6 | 0 | 3 | 63 | 0 | 0 | 0.002373963 |
| 7 | 21 | 1 | 67 | 1 | 0 | 0.8000016 |
| 8 | 61 | 1 | 28 | 0 | 4 | 0.8000016 |
| 9 | 0 | 3 | 63 | 0 | 0 | 0.002373963 |
| 10 | 0 | 1 | 24 | 0 | 2 | 0.08889048 |

## 4.3.2 Selection

This study used Stochastic Universal Sampling (SUS) as selection method. It is implemented by obtaining N equally spaces pointers by generating single random number between [0, AF] as pointer1, and then adding (AF) to generate next pointers, and so on. Where N is the number of required selections, and AF is the average of fitness value in the population. The individual who has a fitness value that spans the positions of the pointers is selected. Table (4.15) shows the selected rules after applying SUS selection on the rules shown in table (4.14).

**Table (4.15): the selected rules from Rootkit attack using SUS selection (Actual data)**

| Selected individual | F1 | F2 | F3 | F10 | F16 | Fitness value |
|---|---|---|---|---|---|---|
| 1 | 708 | 1 | 28 | 0 | 7 | 0.8000016 |
| 1 | 708 | 1 | 28 | 0 | 7 | 0.8000016 |
| 2 | 98 | 1 | 28 | 1 | 14 | 0.8000016 |
| 2 | 98 | 1 | 28 | 1 | 14 | 0.8000016 |
| 5 | 60 | 1 | 28 | 0 | 0 | 0.3200032 |
| 7 | 21 | 1 | 67 | 1 | 0 | 0.8000016 |
| 7 | 21 | 1 | 67 | 1 | 0 | 0.8000016 |
| 8 | 61 | 1 | 28 | 0 | 4 | 0.8000016 |
| 8 | 61 | 1 | 28 | 0 | 4 | 0.8000016 |
| 10 | 0 | 1 | 24 | 0 | 2 | 0.08889048 |

## 4.3.3 Crossover

Uniform crossover operator is used as a crossover method in this research. It is implemented by randomly exchange genes between two parents where the offspring will have 50% of the first parent's genes and another 50% from the second parent's genes. For example:

| | | | | | |
|---|---|---|---|---|---|
| Parent1: | 60 | 1 | 28 | 0 | 0 |
| Parent2: | 21 | 1 | 67 | 1 | 0 |

The result after applying Uniform crossover is:

| | | | | | |
|---|---|---|---|---|---|
| Offspring1: | 21 | 1 | 67 | 0 | 0 |
| Offspring2: | 60 | 1 | 28 | 1 | 0 |

Table (4.16) shows the results of applying Uniform crossover over Rootkit attack rules which shown in table (4.15).

**Table (4.16): results of applying Uniform crossover over Rootkit attack dataset (actual data)**

| ID | F1 | F2 | F3 | F10 | F16 |
|---|---|---|---|---|---|
| 1 | 708 | 1 | 28 | 0 | 7 |
| 2 | 708 | 1 | 28 | 0 | 7 |
| 3 | 98 | 1 | 28 | 1 | 14 |
| 4 | 98 | 1 | 28 | 1 | 14 |
| 5 | 21 | 1 | 67 | 0 | 0 |
| 6 | 60 | 1 | 28 | 1 | 0 |
| 7 | 21 | 1 | 67 | 1 | 0 |
| 8 | 61 | 1 | 28 | 0 | 4 |
| 9 | 0 | 1 | 28 | 0 | 4 |
| 10 | 61 | 1 | 24 | 0 | 2 |

## 4.3.4 Mutation

Flip bit mutation operator is used as a mutation method; it is performed by randomly selecting gene and makes its value equal to a random number of specific range. The probability of mutation rate used in the experiments is 10%, one individual among every 10 will undergo a mutation process. For example, the form of the individual number 1 in table (4.16) was:

| | | | | |
|---|---|---|---|---|
| 708 | 1 | 28 | 0 | 7 |

After applying Flip bit mutation, it became:

| | | | | |
|---|---|---|---|---|
| 708 | 1 | 24 | 0 | 7 |

### 4.3.5 Replacement

Replacement process is used only in SSGA based IDS model. Triple Tournament Replacement (TTR), used in this research, is implemented by comparing three generations among each other, while the rules with highest fitness values will be selected and stored in the rules pool. Table (4.17) shows the results of applying TTR on Neptune attack rules, which has a population size of 10 as a sample size.

**Table (4.17): results of applying TTR over Neptune attack individuals (real data)**

| ID | Generation1 | Generation2 | Generation3 | The Selected individual |
|---|---|---|---|---|
| 1 | 0.80050645 | 0.79869176 | **0.81950756** | 0.81950756 |
| 2 | **0.80050645** | 0.79737413 | 0.79840875 | 0.80050645 |
| 3 | **0.80008414** | 0.79593263 | 0.79840875 | 0.80008414 |
| 4 | 0.39512542 | 0.79593263 | **0.80070015** | 0.80070015 |
| 5 | **0.80050804** | 0.76335016 | 0.79940858 | 0.80050804 |
| 6 | **0.80052233** | 0.80052233 | 0.47503584 | 0.80052233 |
| 7 | 0.75096292 | 0.55843970 | **0.78173195** | 0.78173195 |
| 8 | 0.79886731 | 0.79864659 | **0.79910390** | 0.79910390 |
| 9 | 0.76049135 | 0.74352953 | **0.81950756** | 0.81950756 |
| 10 | 0.75439049 | 0.79910390 | **0.80107165** | 0.80107165 |

## 4.4 Testing Phase and Experimental Results

In this study, the researcher conducted two types of experiments; on the first type, SGA was used for obtaining rules in 20 sub attack types. Then, he tested these rules with testing dataset. On the second type, the researcher applied the same method as in the first experiment but used SSGA instead of SGA. The parameters used in the experiments are presented in the table (4.18).

**Table (4.18): The parameters used in the experiments**

| | |
|---|---|
| Population size | 250 |
| Features encoding | Real number encoding |
| Fitness function | Support Confidence Framework |
| Selection method | Stochastic Universal Sampling (SUS) |
| Crossover operator | Uniform crossover (crossover rate =50%) |
| Mutation operator | Flip bit mutation (mutation rate = 10%) |
| Replacement method* | Triple Tournament Replacement (TTR) |
| Stopping condition | When there are no new rules to be generated |

* TTR used only with SSGA experiments

### 4.4.1 Experimental Results for SGA based IDS

Tables (4.19), (4.20), (4.21) and (4.22), show experimental results from SGA based IDS, which present the number of new generated rules, the training time for each experiment, the DR and FPR for each sub attack type when being tested with the testing dataset. Table (4.19) and table (4.20) illustrate the experimental results for DoS attack types and Probe attack types, respectively. While table (4.21) and able (4.22) show results of U2R attack types and R2L attack types, respectively.

**Table (4.19): Experimental results for DoS attack types using SGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Back | 89 | 00:01:44 | 100% | 0% |
| Land | 0 | 00:00:02 | 57% | 0% |
| Pod | 20 | 00:00:13 | 97.5% | 0.16% |
| Smurf | 41 | 00:00:23 | 56.8% | 0% |
| Teardrop | 2 | 00:00:05 | 100% | 0.37% |
| Neptune | 3885 | 02:18:13 | 96.09% | 1.38% |

**Table (4.20): Experimental results for Probe attack types using SGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Ipsweep | 3187 | 00:17:54 | 98.58% | 6.41% |
| Nmap | 3977 | 00:21:11 | 100.00% | 5.75% |
| Portsweep | 64890 | 04:29:05 | 90.45% | 2.26% |
| Satan | 55873 | 02:43:10 | 91.02% | 9.49% |

**Table (4.21): Experimental results for U2R attack types using SGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Buffer_overflow | 7502 | 00:30:55 | 100% | 3.27% |
| Loadmodule | 34 | 00:00:24 | 100% | 3.23% |
| Perl | 0 | 00:00:02 | 0% | 0% |
| Rootkit | 126 | 00:01:12 | 15.38% | 0.64% |

**Table (4.22): experimental results for R2L attack types using SGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| ftp_write | 10 | 00:00:09 | 0.00% | 0.23% |
| Guess_passwd | 55490 | 10:30:40 | 89.03% | 0.01% |
| Imap | 7902 | 00:36:56 | 100.00% | 0.00% |
| Multihop | 67 | 00:00:50 | 11.11% | 1.52% |
| Phf | 1 | 00:00:03 | 0.00% | 0.00% |
| Warezmaster | 36613 | 09:25:50 | 87.82% | 2.63% |

## 4.4.2 Experimental Results for SSGA based IDS

As mentioned earlier in section (4.4) on the second type of experiments, SSGA is applied on training dataset for generating new rule in order to detect attacks from testing dataset. Tables (4.23), (4.24), (4.25), and (4.26) present results of SSGA based IDS for each attack type. Table (4.23) and table (4.24) illustrate the experimental results for DoS attack types and Probe attack types, respectively. Whereas table (4.25) and table (4.26) show results of U2R attack types and R2L attack types, respectively.

**Table (4.23): experimental results for DoS attack types using SSGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Back | 28 | 00:24:28 | 98.33% | 0.00% |
| Land | 0 | 00:00:03 | 57.14% | 0.00% |
| Pod | 14 | 00:02:56 | 97.56% | 0.16% |
| Smurf | 22 | 00:54:43 | 56.84% | 0.00% |
| Teardrop | 2 | 00:06:00 | 100.00% | 0.37% |
| Neptune | 1276 | 21:55:10 | 95.71% | 1.38% |

**Table (4.24): experimental results for Probe attack types using SSGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Ipsweep | 2074 | 03:30:22 | 98.58% | 3.65% |
| Nmap | 402 | 03:40:11 | 86.30% | 4.98% |
| Portsweep | 27203 | 10:01:55 | 86.62% | 3.62% |
| Satan | 31406 | 11:57:36 | 73.74% | 6.06% |

**Table (4.25): experimental results for U2R attack types using SSGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| Buffer_overflow | 402 | 00:42:01 | 35.00% | 3.22% |
| Loadmodule | 34 | 00:00:58 | 100.00% | 3.23% |
| Perl | 0 | 00:00:01 | 0.00% | 0.00% |
| Rootkit | 88 | 00:03:00 | 0.00% | 0.32% |

**Table (4.26): experimental results for R2L attack types using SSGA based IDS**

| Attack name | No. of new generated rules | Training time | DR | FPR |
|---|---|---|---|---|
| ftp_write | 10 | 00:00:16 | 0.00% | 0.23% |
| Guess_passwd | 19982 | 14:11:22 | 75.39% | 0.01% |
| Imap | 244 | 00:14:31 | 0.00% | 0.00% |
| Multihop | 4 | 00:00:05 | 5.56% | 1.52% |
| Phf | 0 | 00:00:03 | 0.00% | 0.00% |
| Warezmaster | 19390 | 18:34:35 | 79.56% | 2.76% |

After combining the attacks rules of the same category together, and testing each category with the testing dataset, we got the results as shown in table (4.27) and table (4.28).

**Table (4.27): experimental results for Dos, Probe, U2R and R2L categories using SGA based IDS**

| Attack name | DR | FPR |
|---|---|---|
| Dos | 91.78% | 1.92% |
| Probe | 93.58% | 12.14% |
| U2R | 81.08% | 3.92% |
| R2l | 87.54% | 2.86% |

**Table (4.28): experimental results for Dos, Probe, U2R and R2L categories using SSGA based IDS**

| Attack name | DR | FPR |
|---|---|---|
| Dos | 91.36% | 1.92% |
| Probe | 81.83% | 10.19% |
| U2R | 40.54% | 3.53% |
| R2L | 76.40% | 3.00% |

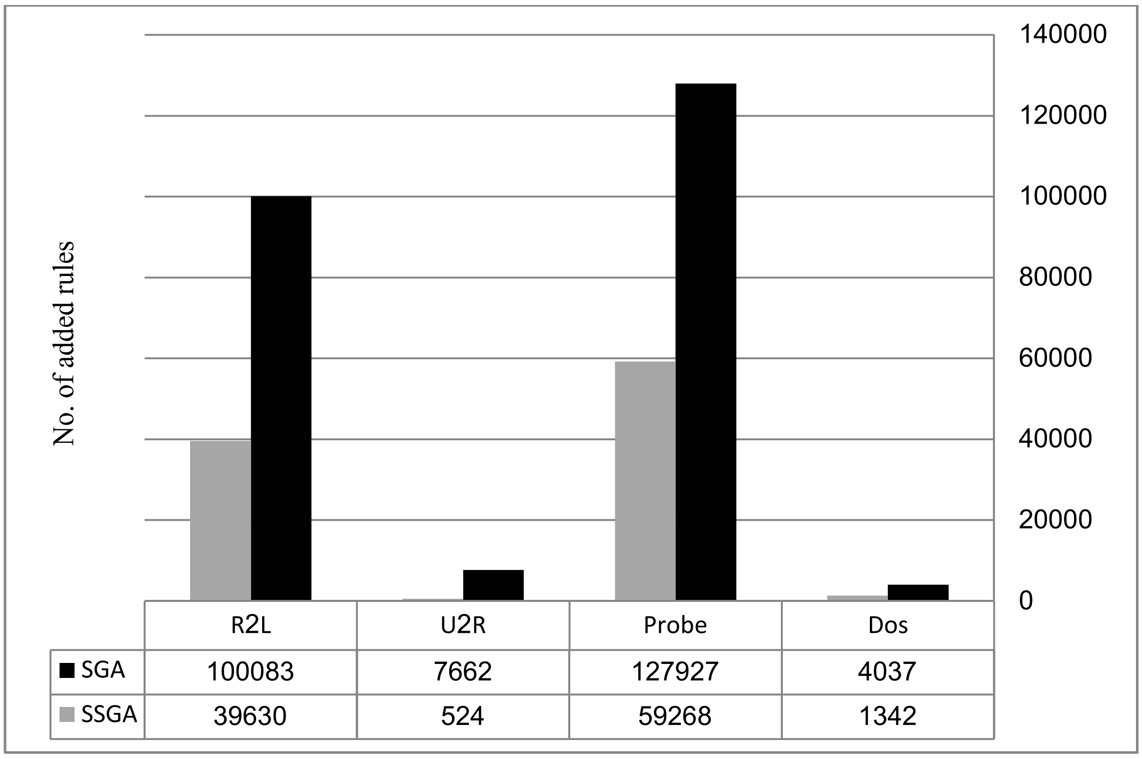
## 4.5 Results Discussion and Analysis

After performing the experiments using both SGA based IDS and SSGA based IDS, and getting the results shown in previous tables. The researcher concluded the following:

1. The generated rules during Genetic Algorithm phase have two cases. In the first case, these rules were already existent in the training dataset and there was no need to store them. In the second case, the generated rules were the new rules and they

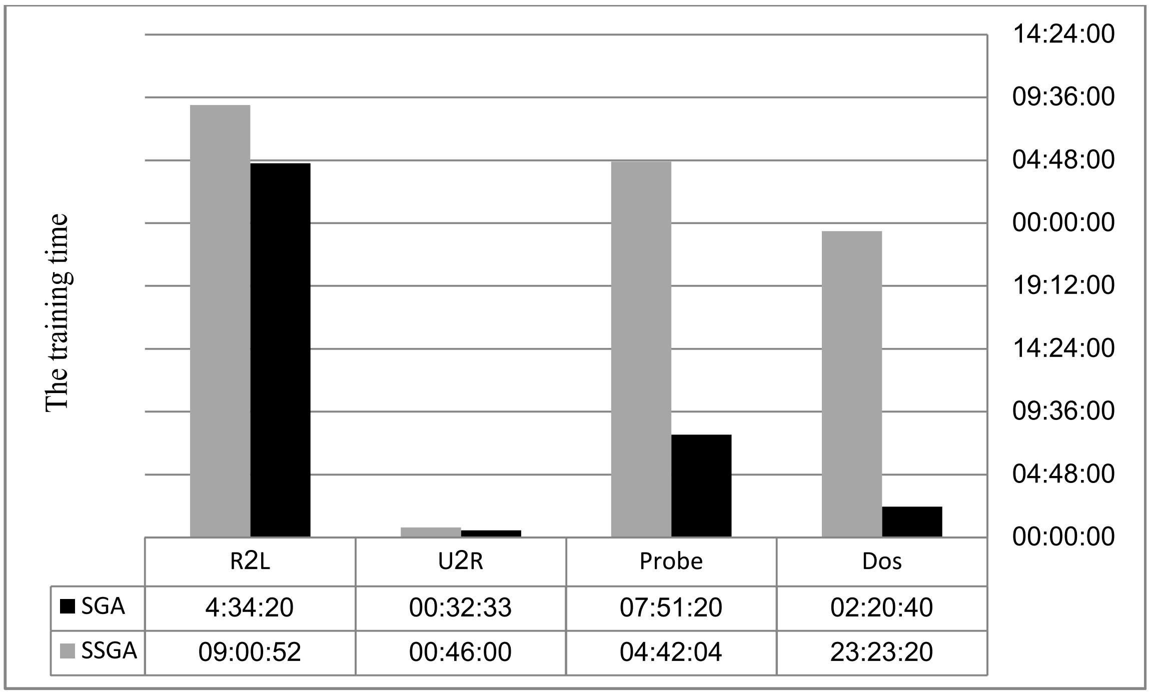
were not existent in the training dataset. According to Support Confidence Framework, the fitness value for these new rules is equal to zero because the value of (A and B) is equal to zero.

2. The new rules have fitness value equal to zero, as mentioned above. Therefore, the Replacement process is unhelpful process because it compares either between already existent rule and new rule (therefore the already existent rule wins because it has a fitness value more than zero) or it compares between new rules, and as in this case where coincidence does play its role in the selection of rules which is stored in the rules pool.
3. The number of added rules using SGA are more than the added rules using SSGA, as shown in figure (4.5), because the new generated rules using SGA will be directly stored in the rules pool at first time, unlike the new generated rules using SSGA that couldn't be stored in the rules pool at first time and could be generated more than once or might not be even stored because of the Replacement process. For the same reason, the training time for SSGA takes longer than the training time for SGA, as shown in figure (4.6).



| | R2L | U2R | Probe | Dos |
|---|---|---|---|---|
| ■ SGA | 100083 | 7662 | 127927 | 4037 |
| ▨ SSGA | 39630 | 524 | 59268 | 1342 |

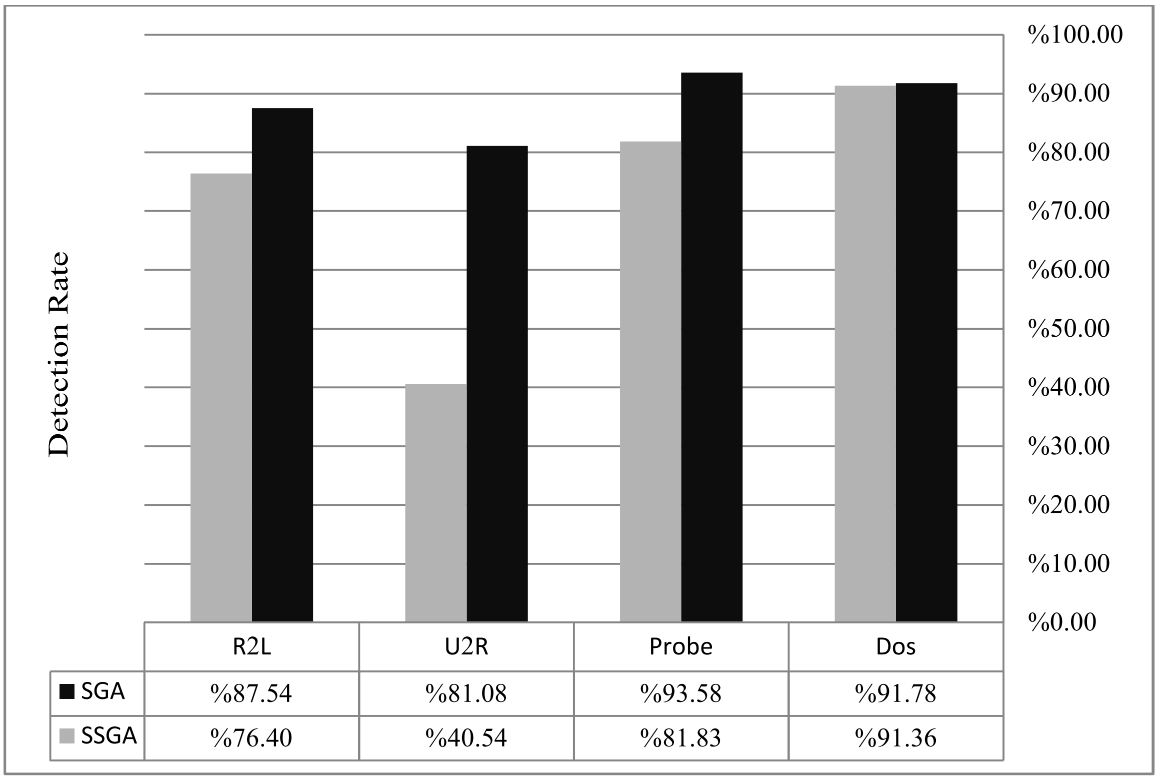
**Figure (4.5): summary of added rules comparison**



| | R2L | U2R | Probe | Dos |
|---|---|---|---|---|
| ■ SGA | 4:34:20 | 00:32:33 | 07:51:20 | 02:20:40 |
| ▨ SSGA | 09:00:52 | 00:46:00 | 04:42:04 | 23:23:20 |

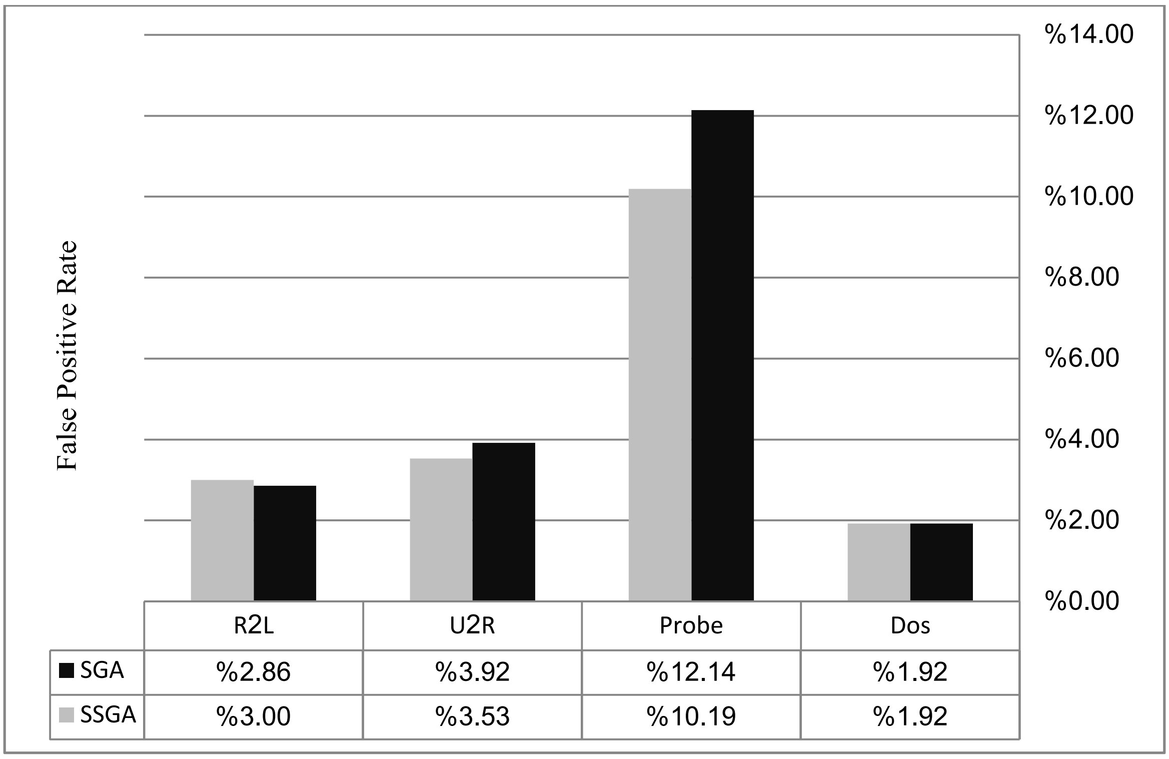
**Figure (4.6): summary of training time comparison**

4. The experimental results demonstrated that SGA based IDS produced more accurate results than SSGA based IDS according to the value of DR, as mentioned in figure (4.7). This is another proof that indicates Replacement process is unhelpful process in intrusion detection field.
5. According to the results of FPR, SGA based IDS achieved higher result than SSGA based IDS in R2L attacks, but it got lower results than SSGA based IDS in Probe and U2R attacks, both achieved equal result in DoS attacks, as shown in figure (4.8).



| | R2L | U2R | Probe | Dos |
|---|---|---|---|---|
| ■ SGA | %87.54 | %81.08 | %93.58 | %91.78 |
| ▨ SSGA | %76.40 | %40.54 | %81.83 | %91.36 |

**Figure (4.7): summary of DR comparison**



| | R2L | U2R | Probe | Dos |
|---|---|---|---|---|
| ■ SGA | %2.86 | %3.92 | %12.14 | %1.92 |
| ▨ SSGA | %3.00 | %3.53 | %10.19 | %1.92 |

**Figure (4.8): summary of FPR comparison**

## 4.6 Comparing the Proposed Study Results with Other Studies

SGA based IDS was compared with other Intrusion Detection System that used Genetic algorithm, the criteria used for comparison are the average of the DR and the Average of the FPR.

Hoque, et al, (2012) presented and implemented IDS by applying SGA. The authors used standard KDD99 benchmark dataset and obtained the average of DR which is equal to 95%, and the average of FPR that is equal to 30.46%.

Aziz, (2014) proposed IDS using SSGA. She also used KDD99 dataset as an environment to train and test the system, and got the average of DR which is equal to 88.25%, and the average of FPR that is equal to 1.48%.

As shown in table (4.29), the average of DR of this research achieved lower results than (Hoque, et al, 2012) and higher results than (Aziz, 2014). On the other hand, the average of FPR of this research achieved higher results than (Hoque, et al, 2012), but lower than (Aziz, 2014).

**Table (4.29): comparison results of this research with other results**

| | Average of DR | Average of FPR |
|---|---|---|
| This research | 88.5% | 5.21% |
| Hoque, et al, (2012) | 95% | 30.46% |
| Aziz, (2014) | 88.25% | 1.48% |

# Chapter Five

# Conclusion & Future Work

## 5.1 Conclusion

This study proposed two models of Intrusion Detection Systems to detect network intrusions. In the first model, an Intrusion Detection System is built using Simple Genetic Algorithm (SGA based IDS). While the second model, an Intrusion Detection System is built using Steady State Genetic Algorithm (SSGA based IDS). These models were implemented and evaluated using NSL-KDD intrusion detection dataset.

In order to determine which feature set is the most suitable one for each attack category, the author compared published studies regarding this topic and modeled a hybrid feature set containing the best available feature sets.

The training process showed that the training time for SSGA was much more than the time required for SGA, despite that, the numbers of new generated rules using SGA are more than those using SSGA.

SGA based IDS achieved an average of DR equal to 88.5%, while SSGA based IDS achieved an average of DR equal to 72.53%. On other hand, SGA based IDS produced an average of FPR equal to 5.21%, while SSGA based IDS produced an average of FPR equal to 4.66%.

From the experimental results, the researcher concluded that performing an IDS using SGA gives higher performance results than using SSGA according to the value of

DR and the number of new generated rules, also the training time for SGA experiments is shorter than the training time for SSGA. On other hand, SSGA based IDS achieved an FPR average that was relatively better than SGA based IDS.

The results of DR and FPR were compared with results of other researches, and showed that the results of this research was convincing.

## 5.2 Future Work

1. More comprehensive researches are needed to determine key features for every attack category in order to build an efficient and an effective IDS.
2. Additional studies are needed to compare Genetic Algorithm with other evolutionary algorithms in Intrusion Detection field.
3. Comparing SGA and SSGA using anomaly based IDS.
4. Combining Genetic Algorithm and fuzzy logic to improve the accuracy of IDS.
5. Developing an Intrusion Detection System that is able to detect Misuse and Anomaly attacks with high DR and low FPR.

## References

- Abdullah, B., Abd-Alghafar, I., Salama, G. I., and Abd-Alhafez, A. (2009). Performance evaluation of a genetic algorithm based approach to network intrusion detection system", ***13th international conference on aerospace sciences and aviation technology,*** May 26 – 28, Military Technical College, Kobry Elkobbah, Cairo, Egypt.

- Adewumi, A.O. (2010). **Some improved genetic-algorithms based heuristics for global optimization with innivative applications**, (Master thesis). University of the witwatersrand. Johannesburg. South Africa.

- Alabsi, F. (2012). **An Enhanced Steady State Genetic Algorithm Model for Misuse Network Intrusion Detection System**, (master thesis). Middle East University, Amman, Jordan.

- Al-Sharafat, W. S., and Naoum, R. (2009). Development of Genetic-based Machine Learning for Network Intrusion Detection. ***World Academy of Science, Engineering and Technology,*** Vol. 3**.**

- Al-Zokari, Y. I. (2008). **Computer intrusion detection system via pattern recognition technique**, (unpublished master thesis). The University of Jordan, Amman, Jordan.

- Amiri, F., Yousefi, M., Lucas, C., Shakery, A., and Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. ***Journal of Network and Computer Applications***, Vol. *34*, Issue 4, pp 1184-1199.

- Aziz, S. (2014), **An Enhancement of the Replacement Steady State Genetic Algorithm for Intrusion Detection**, (master thesis). Middle East University, Amman, Jordan.

- Chou, T. S., Yen, K. K., Luo, J. (2008). Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms. ***International journal of computational intelligence***, Vol.2, No.11, pp. 196-208.

- Chowdhary, M., Suri, S., and Bhutani, M. (2014). Comparative Study of Intrusion Detection System. ***International Journal of Computer Science and Engineering***, Vol.2, Issue 4, ISSN: 2347-2693.

- Crosbie, M., and Spafford, G. (1995). Applying genetic programming to intrusion detection. *In Working Notes for the AAAI Symposium on Genetic Programming* (pp. 1-8). MIT, Cambridge, MA, USA: AAAI.

- Dhak, B. S., and Lade, S. (2012). An evolutionary approach to intrusion detection system using genetic algorithm. ***International Journal of Emerging Technology and Advanced Engineering***, Vol. *2*, Issue 12, ISSN: 2250-2459.

- Duran, B., Xhafa, F. (2006). The effects of two replacement strategies on a genetic algorithm for scheduling jobs on computational grids. ***In Proceedings of the 2006 ACM symposium on Applied computing***, (pp. 960-961). ACM.

- French, T. R. (2012). *Evolutionary optimisation of network flow plans for emergency movement in the built environment*.

- Ghosh, P., and Mitra, R. (2015). Proposed GA-BFSS and logistic regression based intrusion detection system. In *Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on* (pp. 1-6). IEEE.

- Goyal, A., and Kumar, C. (2008). GA-NIDS: a genetic algorithm based network intrusion detection system. *Northwestern university*.

- Goyal, M. K., Aggarwal, A., and Jain, N. (2012). Effect of Change in Rate of Genetic Algorithm Operator on Composition of Signatures for MisuseIntrusion Detection System. In *Parallel Distributed and Grid Computing (PDGC),* ***2012 2nd IEEE International Conference on*** **(pp. 669-672). IEEE.**

- Guo, P., Wang, X., and Han, Y. (2010). The enhanced genetic algorithms for the optimization design. *3rd International Conference on Biomedical Engineering and Informatics (BMEI),* Vol. 7, pp. 2990-2994. IEEE.

- Hasan, B. H., and Mustafa, M. S. (2011, January). Comparative Study of Mutation Operators on the Behavior of Genetic Algorithms Applied to Non-deterministic Polynomial (NP) Problems. *Second International Conference on Intelligent Systems, Modelling and Simulation,* (pp. 7-12). IEEE.

- Hoque M. S., Mukit A. and Bikas A. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm, ***International Journal of Network Security and its applications,*** Vol.4, NO.2, 109-120.

- Hu, X. B., and Di Paolo, E. (2009). An efficient genetic algorithm with uniform crossover for air traffic control. *Computers & Operations Research*, *36*(1), 245-259.
- Jongsuebsuk, P., Wattanapongsakorn, N., and Charnsripinyo, C. (2013). Network intrusion detection with Fuzzy Genetic Algorithm for unknown attacks. *International Conference on Information Networking (ICOIN),* (pp. 1-5). IEEE.

- Jones, J., Soule, T. (2006). Comparing genetic robustness in generational vs. steady state evolutionary algorithms. In *Proceedings of the 8th annual conference on genetic and evolutionary computation* (pp. 143-150). ACM.

- Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2005). Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. ***In Proceedings of the third annual conference on privacy, security and trust.***

- Kumar, M., Husian, M., Upreti, Nand Gupta, D. (2010). Genetic algorithm: review and application. ***International Journal of Information Technology and Knowledge Management***, Vol.2, No.2. PP. 451-454.

- Lu, W., and Traore, I. (2004). Detecting new forms of network intrusion using genetic programming. ***Computational Intelligence***, 20(3), 475-494.

- Mehra, M., Jayalal, M. L., Arul, A. J., Rajeswari, S., Kuriakose, K., and Murty, S. S. (2012). Design and Development of Genetic Algorithm for Test Interval Optimization of Safety Critical System for a Nuclear Power Plant. In *Online Proceedings on Trends in Innovative Computing, Intelligent Systems Design and Applications Conference, Kochi, India.*

- Mukkamala, S., Sung, A., Abrham, A., (2004), Modeling Intrusion Detection System using Linear Genetic Programming Approach, Proceeding IEA/AIE 17th International Conference on Innovations in Applied Artificial Intelligence, PP 633-642, ISBN: 3-540-22007-0.

- Naoum, R., Aziz, S., and Alabsi, F., (2014). An Enhancement of the Replacement Steady State Genetic Algorithm for Intrusion Detection. ***International Journal of Advanced Computer Research,*** Vol.4, No.2, Issue 15, ISSN: 2249-7277.

- NSL-KDD dataset. Available on: http://nsl.cs.unb.ca/NSL-KDD/

- Ojugo, A. A., Eboka, A. O., Okonta, O. E., Yoro, R. E., and Aghware, F. O. (2012). Genetic algorithm rule-based intrusion detection system (GAIDS). ***Journal of Emerging Trends in Computing and Information Sciences***, Vol.3, No.8, ISSN: 2079-8407, pp. 1182-1194.

- Pal, D., and Parashar, A. (2014). Improved genetic algorithm for intrusion detection system. In *Computational Intelligence and Communication Networks (CICN), 2014 International Conference on* (pp. 835-839). IEEE.

- Pencheva, T., Atanassov, K. and Shannon, A. (2009). Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets. In *Proceedings of the Tenth International Workshop on Generalized Nets, Sofia* (pp. 1-7).

- Prasad, K. K., Borah, S. (2013). Use of Genetic Algorithms in Intrusion Detection Systems: An Analysis. ***International Journal of Applied Research and Studies (iJARS)***, Vol.2, Issue 8, ISSN: 2278-9480.

- Jebur, S. A., & Nasereddin, H. O. (2015). Enhanced Solutions for Misuse Network Intrusion Detection System using SGA and SSGA. ***IJCSNS International Journal of Computer Science and Network Security,*** Vol.15, No.5, ISSN: 1738-7906.

- Selvakani, S., and Rajesh, R. S. (2007). Genetic Algorithm for framing rules for Intrusion Detection. ***IJCSNS International Journal of Computer Science and Network Security***, 7(11), 285-290.

- Sharma, D., Singh, V., & Sharma, C. (2012). GA Based Scheduling of FMS Using Roulette Wheel Selection Process. ***In Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011), Springer India,*** pp. 931-940.

- Shaveta, E., Bhandari, A., and Saluja, K. K. (2014). Applying Genetic Algorithm in Intrusion Detection System: A Comprehensive Review. ***Association of Computer Electronics and Electrical Engineers.***

- Shirazi, H. M., Kalaji, Y., (2010). An intelligent intrusion detection system using genetic algorithms and features selection. ***Majlesi Journal of Electrical Engineering***, Vol. 4, No. 1.

- Shiri, F. I., Shanmugam, B., and Idris, N. B. (201). A parallel technique for improving the performance of signature-based network intrusion detection system. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on* (pp. 692-696). IEEE.

- Soon, G. K., Guan, T. T., On, C. K., Alfred, R., and Anthony, P. (2013). A comparison on the performance of crossover techniques in video game. ***IEEE International Conference on Control System, Computing and Engineering***, 29 Nov. -1 Dec. 2013, Penang, Malaysia. IEEE.

- Stallings, W. and Brown, L. (2008). *Computer security: principles and practice*, (second edition), Upper Saddle River, NJ: Prentice Hall.

- Tavallaee, M., Bagheri, E., Lu W. and Ghorbani, A. (2009). A detailed analysis of the KDD CUP 99 data set. ***Proceedings of the 2009 IEEE symposium on computational intelligence in security and defense applications (CISDA 2009).***

- Torkaman, A., Javadzadeh, G., and Bahrololum, M. (2013). A hybrid intelligent HIDS model using two-layer genetic algorithm and neural network. *5th Conference on Information and Knowledge Technology (IKT)*, (pp. 92-96). IEEE.

- Wong, M. L., and Leung, K. S. (2000). *Data mining using grammar based genetic programming and applications*. Netherlands, Kluwer Academic Publishers.

- Zhao, J. L., Zhao, J. F., and Li, J. J. (2005). Intrusion detection based on clustering genetic algorithm. In *Machine Learning and Cybernetics. Proceedings of International Conference on* (Vol. 6, pp. 3911-3914). IEEE.

- Zhou, H., Meng, X., and Zhang, L. (2007). Application of support vector machine and genetic algorithm to network intrusion detection. In Wireless Communications, Networking and Mobile Computing, WiCom 2007. ***International Conference on IEEE. (pp. 2267-2269).***