# Implementation of Genetic Algorithm for Rule Generation in IDS

*Saurav Poudel*

*Department of Science*

*Edith Cowan University*

*Email:* [*spoudel2@our.ecu.edu.au*](mailto:spoudel2@our.ecu.edu.au)

**Abstract**

*This paper describes a technique to generate rules for intrusion detection system using a genetic approach. An overview of the components is presented along with previous work on this domain. An analysis of the existing dataset has been done and a replacement is proposed along with optimization technique through condensation of the dataset and feature selection process have been discussed. Unlike other implementation, this model incorporates a new feature called the difficulty level of the data which will be very helpful to identify the attacks that were previously very complex to analyze and difficult to identify.*

**Keywords** *Intrusion Detection System, Genetic Algorithm, Rule Based IDS,KDDCup99 Dataset.*

## Introduction

The internet is evolving and with it so does the security threats. As network infrastructures are scaling exponentially, the need for efficient security measures like Intrusion detection systems (IDS) is also increasing. IDS is one of the best lines of defense in a security system. They provide appropriate methods to prepare for and deal with attacks. These systems are designed to detect anomaly in the system and alert the users.

The dynamic nature of the network makes it hard to detect anomalous behavior. Effective IDS systems use rules to detect intrusion. These rules also called the signature are derived by analyzing the patterns of previously known attacks. If a similar type of traffic is encountered, it is labeled as an intrusion. Legacy systems heavily relied on human experts to generate these rules. They did so by monitoring networking consoles and analyzing audit logs. This is a very time consuming and inefficient method; besides they don’t scale well. An efficient way of generating rules for IDS systems is the use of machine learning. Machine Learning uses adaptive learning methods to analyze big data and generate meaningful information. Genetic Algorithm is a branch of machine learning which solves optimization problems very efficiently.

IDS was a hot research topic in the early and late 2000’s but the interest has been gradually decreasing because of the outdated network datasets which are used to generate the rules. This research study focuses on the use of Genetic algorithm for rule generation, critical analysis of the existing dataset and exploring new optimization criterion.

In order to understand the way in which a ruleset can be generated using a genetic approach; the information in the following chapter describes the key elements needed for understanding the structural approach of the research. An introduction to the Intrusion detection system, Genetic algorithm, and its components will be discussed. Finally, after a review on related works, an optimized algorithmic solution for rule generation in IDS will be proposed.

**Intrusion Detection Systems**

Intrusion Detection System (IDS) is an automated system which provides continual vigilance of network traffic in real time to preserve the integrity of the information and maintain system availability during an intrusion. When a user of an information system takes an action that the user was not legally allowed to take, it is called intrusion (Jones and Sielken, 2000). These intruders can either have access permission or not based on which, they are classified as internal and external intruders respectively. IDS are devices designed to successfully and accurately detect and report the intrusions. These systems are usually specific to the operating system that they operate in and are an important tool in the overall implementation of an organization’s information security policy which reflects an organization’s statement by defining the rules and practices to provide security, handle intrusions and recover from damage caused by security breaches (Wei Li, 2004).

Denning proposed an approach to determine computer misuse using Real Time Intrusion Detection System in 1987 (Denning, 1987). Before the introduction of such systems, human experts have been responsible for determining intrusions by monitoring networking consoles and auditing logs. Since then various approaches to automate the detection system have been proposed. Artificial Intelligence is one of the most efficient implementation methods of IDS in recent years. IDS was the hottest research area in Computer Security during the early and late 2000s. The loss of excitement is mostly because of the lack of research in new network data set used to train such systems.

Anomaly Detection and Misuse Detection are the two basic approaches for detecting intrusions. The first approach defines the acceptable dynamic behavior of the current system and any deviation from this behavior is classified as an intrusion while the later classify an attack based on the known ways to penetrate a system. The known ways of penetrating the system are referred to as signature or patterns. Any network behavior that matches the pattern is classified as an intrusion.

Based on the implementation method, IDS are classified into *Network Based IDS (NIDS)* and *Host-Based IDS*. An IDS which monitors traffic through the network devices such as Network Interface Card is known as *NIDS* while Host Based IDS monitors network data related to software environment of a system. A proactive IDS is referred as an *Intrusion Prevention System.*

### Rule Based Intrusion Detection System

Any misuse-based approach of intrusion detection systems needs a certain patterns or signatures which are referred to classify a network packet as misuse or intrusion. These patterns and signatures form the rules for the IDS system. These rules classify the network packets as normal and intrusions. Rules are generally bound by *If-Else* statements as defined below:

If {condition :} then {action}

The condition in these rules are defined by the signature or pattern of the network data. Network data consists of different features. These features include Source and Destination IP address, Port Number, Flags, Services etc. A description of Network Data will be further discussed in chapter below.

The rules stored in the rule base are usually in the following form (Sinclair, Pierce, and Matzner 1999):

If {condition} then {act}

For example, a rule can be defined as:

*If* {

Source IP address 14.13.112.01;

Destination IP address: 151.84.21.12;

Destination port number: 23;

Connection time: 4sec

}

Then {

**Alert**: attack

}

The network packets are audited by the IDS systems using a set of such rules which is called a *ruleset* of an IDS.

**Introduction to Genetic Algorithm**

#### Genetic algorithms (GA) are a family of computational models which mimics the Darwinian principle of evolution as a problem-solving strategy. (Wei Li, 2004). A genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a search space (Whitley, 1994). Similar to how a population evolves to the most successful individuals in the population through reproduction resulting in crossover and mutation, an analogy of such characteristics is used to generate a solution to the real-world problems. In our case; to generate the fittest set of rules which can detect attacks successfully. The success of an individual is determined by fitness rules in the environment (Eiben & Smith, 2003).

#### GA maintains a certain set of individuals in a population. Population size is the number of individuals in a population. These individuals represent the solution space for our problems. Each individual has a value (chromosome composed of genes) and its quality (fitness value). Generations of such a population of individuals are evolved using reproduction. Reproduction is done by mating two individuals in a population and the probability of an individual selected for mating is directly proportional to its quality. Reproduction is performed using crossover and mutation. These new set of individual then replace the old population and start a new generation. In each generation, individuals of best quality mate to evolve into a population of the maximal quality.

##### Representation

An individual is represented by a chromosome. Each chromosome consists of genes. These genes are the digital encoding of a solution space. Binary bits represent the genetic encoding of an organism. Each gene can have a value of 0 or 1. A solution to the problems is represented using a number of required genes.

If let’s say x1,x2,x3… xn be the features of a Network data. A numeric representation of x1 is defined as the index of the value in the list of all possible values of the feature.

For a network feature “protocol type” which can take values UDP, TCP, and ICMP. Each value can be represented as their indexes 1, 2 & 3.

If a network dataset consists of the same feature with 2 distinct values all over the network, a single binary bit can be used to represent the feature. Meaning a single gene can represent distinct values of the feature:

|  |  |
| --- | --- |
| UDP | 0 |
| TCP | 1 |

*Table1: Binary encoding for network feature “protocol type”*

|  |  |
| --- | --- |
| UDP | 00 |
| TCP | 01 |
| ICMP | 10 |

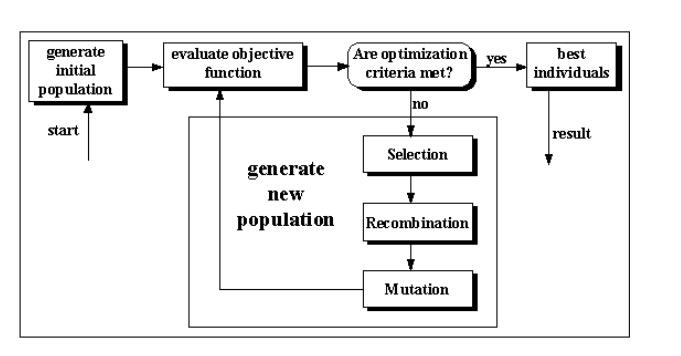
*Table 2: Binary Encoding for features with more values*

The values *UDP, TCP, ICMP* are called *phenotypes* while the binary representation of these values are called *genotypes*. The number of genes required for a genotype representation depends on the range of values it takes. A length of n gene can represent distinct values. In order to represent the service feature which has 66 distinct values 7 genes are required. The number of genes required is the nearest value of n whose power of 2 is just greater or equal to the number

##### The Canonical Genetic Algorithm

The implementation of a Genetic Algorithm starts with a generation of the initial population. Each individual in the population is a binary string which encodes our problem as defined in the section above.

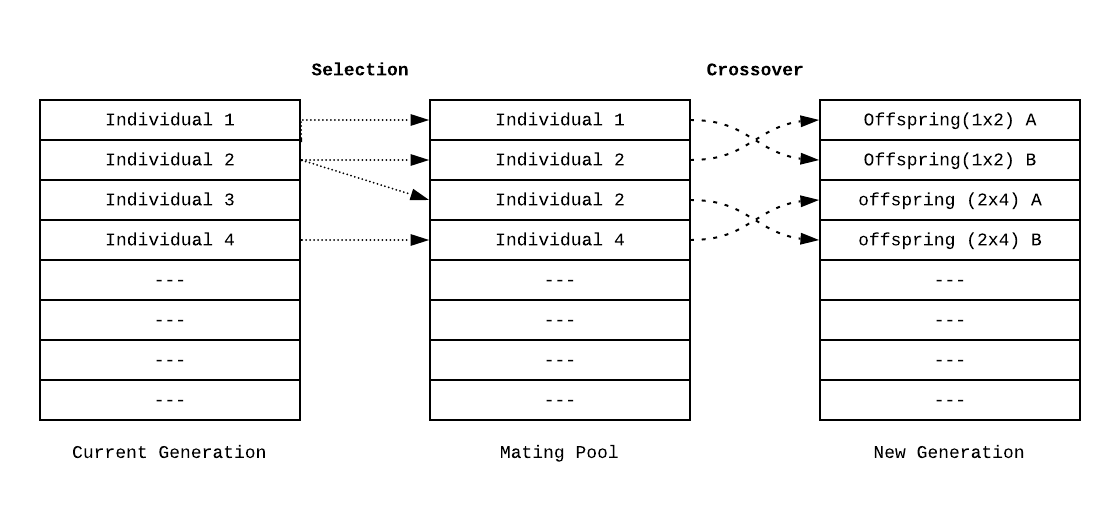
The individual in a population has a quality which is calculated as the fitness of the individual. The fitness value is a measure of performance. It is defined with respect to other individuals in the population and is an indicator of reproductive opportunity. The evaluation function is mathematical equations which correspond to the nature of the data and characteristic of the problem. We use these evaluation function to calculate the fitness of each individual.

A genetic algorithm uses the current population to generate a new population of the individual through a process called reproduction. This transition from the current to the new population starts with a selection process to generate a mating pool. This is an intermediate population which is generated for reproduction. A selection process determines which individuals are selected to reproduce. These selected individuals create the mating pool.

*Fig 1: A simple genetic algorithm (Pohlheim, 2003)*

Individuals from mating pool are selected in random to perform crossover and mutation. The selection of individuals in the mating pool is proportional to their fitness values. There can be multiple copies of a single individual in the mating pool depending in their fitness values.

Suppose two individual*I1* have fitness values of 0.2 and 0.4 respectively. If *I1* is repeated n times in the mating pool *I2* will be repeated 2n times, such that the probability of *I2* being selected for reproduction is double.



*Fig 2: A chart showing the how a new generation is formed from a current generation. An intermediate mating pool is used to select individuals for crossover operation. Crossover creates a child by interchanging genes between two individuals.*

Individuals are selected for reproduction in random from the mating pool. Crossover operation is performed in each individual. A crossover operation generates a sample point in space by combining two parent strings. The selected individuals are called parents. A random index is selected and the bits before are after the index are interchanged between two parents to create a child. If a single point is selected for crossover it is called a *single point crossover*. A *two-point crossover* splits the individuals into two sample spaces and exchanges the genes between the parents.

Consider an individual with chromosome 1101101010 and another individual 1001100110. A single point crossover occurs as follows:

*1101* \/ *101010*

**1001 /\ 100110**

Offspring are produced by interchanging the fragments from the parents as follows:

*1101***100110** and **1001***101010*

This operation is also called recombination (Whitley, 1994). Mutation on each child is performed to bring variation in the population. Randomly flipping a bit with a very low probability (as low as 0.1%) for a certain gene is performed after the crossover process. After completing these processes, a new set of population is evaluated which completes a generation. The number of generations in the genetic algorithm is set by the programmer.

## Literature Review

A genetic algorithm tutorial defining both encoding and optimization problem was presented by Darrell Whitley. The paper successfully defined the optimization of criterions for genetic algorithm based on the types of problems it is used to solve (Whitley, 1994). Some of the earliest work on Genetic algorithm in network security is by Forrest where a theoretical model for securing a digital asset is demonstrated. A change detection method analogous to the human immune system was used to solve the problem of computer virus detection. (Forrest, Perelson, Allen & Cherukuri, 1994).

Li has applied genetic algorithms on both temporal and spatial network connection data to identify anomalous network behaviors (Wei Li, 2004). Because of the absence of a standard data set which incorporates temporal (time) dimension; a statistical measure is not available.

Optimization of the IDS system by improving the fitness function is proposed by Diaz-Gomez (Diaz-Gomez, P.A., & Hougen, D.F, 2005). The paper determines that a system performance improves with a better fitness function even when other parameters remain constant.

A rule set with 80% efficiency was generated by (Ojugo et. al, 2012) parameterizing the number of generation so as not to overtrain the model. Overtraining the data leads to biases towards the training data which is again inefficient to detect network intrusion in the real world scenario.

Chittur Adhitya proposed a model with an overall accuracy of 97.8 (Chittur, 2001). The paper suggested a better fitness function and other parameters will increase the efficiency of the model. The experiments were performed in a small training and testing datasets. This limits the detection rate of rulesets to certain attacks and moreover, such statistical measure might not scale well in a bigger dataset.

A dynamic niche sharing approach of genetic algorithm for a multimodal domain by Miller and Shaw defines methods which enable the algorithm to identify multiple optima within multimodal datasets. Such an approach is suitable for network data (Miller & Shaw, 1995).

A Genetic Algorithm based IDS for Mobile Ad-hoc network has been studied where the behavior of nodes in the network is monitored using a Genetic Approach. The feature in the network like Request Forwarding Rate, receive rate, etc. are used to configure the rules for the IDS system (Sujatha, Dharmar & Bhuvaneswaran, 2012).

An approach called Neighborhood Outlier Approach was implemented to successfully improve the performance of the IDS. (Jabez & Muthukumar, 2015). This approach aims to detect newer attacks successfully by detecting novel data packets and classifying using a threshold value. Another approach to improve IDs performance was done by a reduction in feature using different selection techniques. (Desale & Ade, 2015) A comparative study of techniques like CFS, IG, and CAE to reduce the feature and their effects in detection using an NSL KDD dataset was performed.

A black hole is a packet drop ambush attack performed in a network layer. An evolutionary approach to detect such attacks in a MANET (Mobile Ad-hoc Network) was done by (Thanuja & Umamakeswari, 2018). Using HPSO-GA (Hybrid Particle Swarm Optimization) the approach learns the behavior of the ad-hoc network, uses it to classify the black hole attacks. This approach to detecting a novel attack shows the efficiency of an evolutionary approach to detect attacks. A comparative study on a hybrid approach called GSPSO-ANN (A combination of Gravity Search Particle Swarm Optimization) was studied by Dash (Dash, 2015). He performed experiments in individual approaches i.e. Gravity Search Artificial Neural Network (GS-ANN) and Particle Swarm Optimization Artificial Neural Network (PSO-ANN) and compared it to the hybrid approach. The hybrid approach showed improved performance over popular techniques like DT (Decision tree), GA-ANN (Genetic Algorithm Artificial Neural Network). A fuzzy genetic approach proposed by (Danane & Parvat, 2015) which classifies the network data using a threshold unlike sequential algorithms indicates improvement in detection rate and reduction in training time. A vector-based approach which condenses the feature set and classifies all the attack instances separately also showed an improvement in detection rates (Ijaz, Hashmi, Asghar & Alam, 2017).

Most IDS rely on outdated datasets to train the machine and generate signatures. A rather unconventional method to obtain the training data using a large-scale PHP web application was done by Bronte. (Bronte, Shahriar & Haddad, 2016). The web application data were analyzed to generate a regular expression as a signature to detect attacks like Cross-Site Scripting, SQL Injection, and Remote File Inclusion. This is a Host Based IDS which analyses the data of the network packet rather than network features.

IDS for wireless mesh network where GA is primarily used for feature selection has been done in recent years. (Vijayanand, Devaraj & Kannapiran, 2018). Feature selection is a data mining process where GA is used to find the most significant set of features to analyze and use in a rule base. A study on a similar approach using SVM (Support Vector Machine) to classify the data and GA for parameter setting and feature selection was done. (Gauthama Raman, Somu, Kirthivasan, Liscano & Shankar Sriram, 2017) This approach called HG-GA (HyperGraph Genetic Algorithm) uses a weighted objective function to avoid premature conversion and exploits the hyper-clique property of a hypergraph to generate an optimal initial population. Similarly, an approach to get a better solution by optimizing the initial population called the K-means Initial population strategy (KIP) was proposed by (Tao, Sun & Sun, 2018). This paper determines that the initial population plays an important role in finding the optimal solution.

A comprehensive study on different approaches to intrusion detection system including Evolutionary algorithms like XCS(Extended Classifier System),GASSIST(Genetic Algorithm Based Classifier System),MPLC(Memetic Pittsburgh Classifier System) and non-evolutionary rule learning algorithms like RIPPER(Repeated Incremental Pruning to Produce Error Reduction) and DT (Decision Tree) was done by (Rastegari, S., 2015).To overcome the challenges of such system a new system called ESR-NID (Evolving Statistical Rulesets for Network Intrusion Detection) which utilizes an incremental learning approach to learn the changing behavior of user and attack patterns were proposed. The experiment produced a comparable result to other tested methods.

Datasets which is used to train the models define the efficiency of the IDS system. Better training dataset generates a better rule. To overcome some of the limitations of the existing dataset (Chakrabarty, Chanda & Saiful, 2017) proposed an anomaly-based IDS system using a genetic approach and K-centroid clustering. Biases of rules to the redundant data is a major concern while training a system. The proposed system overcomes the limitation by using clustering techniques. The most prevalent data-set for training and testing such systems is “KDD Cup 1999 Dataset” ("KDD Cup 1999 Data", 1999). A detailed analysis of this dataset performed by (Tavallaee, Bagheri, lu & Ghorbani, 2009) showed the following prevalent characteristics which affect the effectiveness of the IDS system:

* No experimental validation of data’s false alarm characteristics.
* Method of data collection (TCP-dump) become overloaded and usually drop packets. There is no discussion of dropped packets.
* Redundant Records
* Duplicate Records

An improved dataset by Canadian Institute of cybersecurity ("NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB", 2009) has been proposed which has solved some inherent problems mentioned (Tavallaee, Bagheri, lu & Ghorbani, 2009).

The above review of the literature indicates the opportunity to further expand the knowledge in this field by using a novel approach to using genetic algorithms as part of an investigation into IDS. Because of the absence of updated and newer datasets, the model trained in almost all the research lack information. Besides the nature of GA to converge to local maxima results in loss of valuable information. The network data falls under a multimodal domain, developing a GA using niche sharing techniques will provide a better set of rules. Experiments performed by researches shows promising statistical figures but the higher percentage of detection rate is because of the presence of duplicate records and particular attacks having a majority in the sample data. For example, DOS attacks comprise 70% of the test set. Even though there are very few instances of attack types such as “multihop”, “rootkit”, “phf” etc, they are significant intrusion methods which can compromise the CIA in computer security. The detection of such attacks is equally important. The improved dataset from Canadian Institute of Cyber Security has labeled each entry by their difficulty level of detection. These labels will be exploited to generate a rule set which will be able to successfully identify a wide range of attacks.

# Genetic Algorithm Applied to Intrusion Detection

This chapter will provide a solution to rule generation using a canonical genetic algorithm with various optimization techniques. This section justifies the selection of dataset by analyzing the limitations of existing data sources and genetic algorithm as an approach. Challenges in developing an efficient solution will be discussed to finally provide an algorithm solution.

##### Analysis of KDDCUP99 dataset

KDDCup99 is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 the Fifth International Conference on Knowledge Discovery and Data Mining. ("KDD Cup 1999 Data", 1999). The characteristics of the dataset are defined below by referencing the information produced by an open source machine learning platform called “openml” by Vanschoren. (Vanschoren, n.d.).

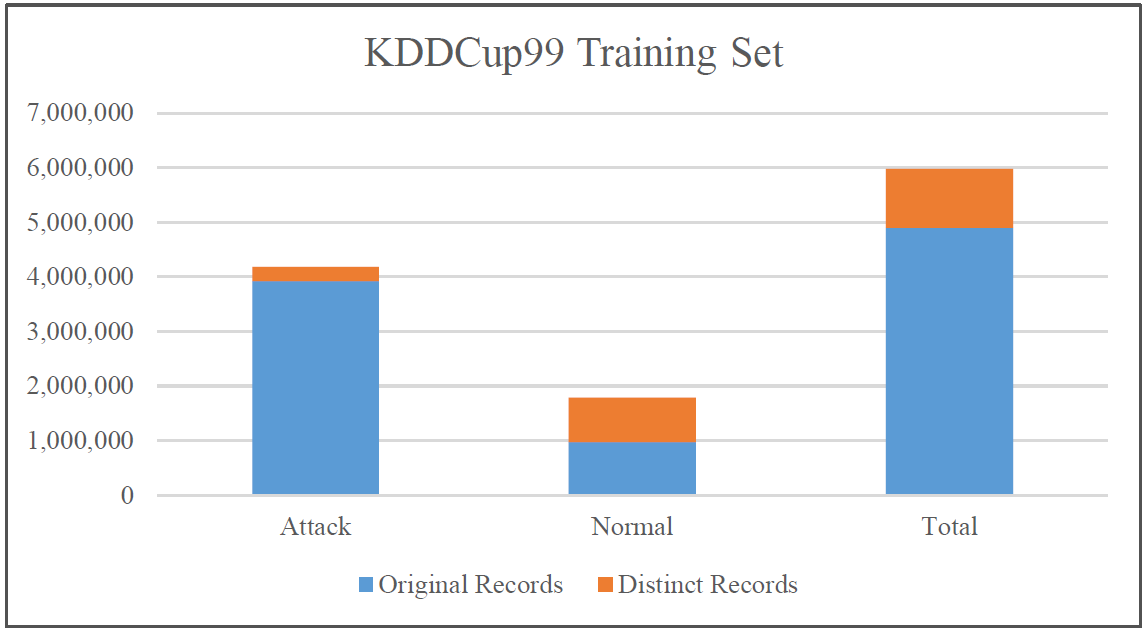
|  |  |
| --- | --- |
| Characteristics | Measurement |
| Number of Features | 42 |
| Total Number of instances | 4898431 |
| Number of classes (*Attack Types*) | 23 |
| Number of normal connections | 972781 |
| Number of attacks | 3,925,650 |

*Table 3: Characteristic measure of KDDcup99 dataset*

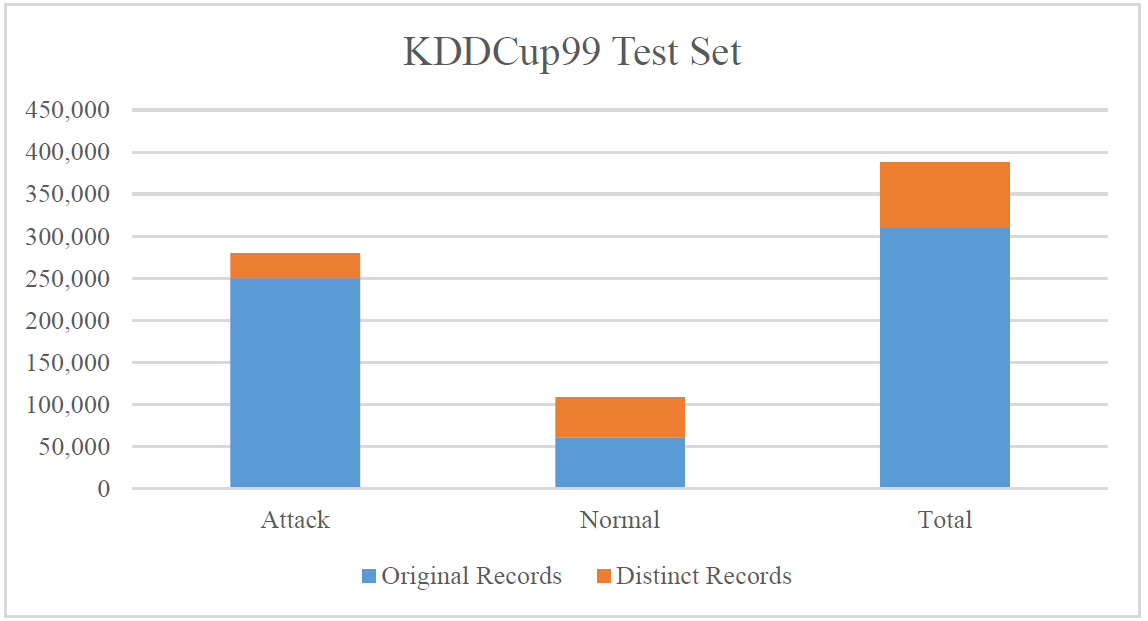
According to the detailed analysis performed by (Tavallaee, et al., 2009), there are numerous inherent problems in this dataset which was briefly discussed in the literature review. The two major subsets of the dataset namely KDD Train Set and KDD Test Set which is used to train the machine and test the results respectively are improved by reducing the redundant records. The distribution of attacks in the dataset is very uneven which results in a very inefficient rule and longer training time. Almost 71% of the training data consists of DoS (Denial of Service) attacks, this results in a rule set which will affect evaluation methods in the Genetic Approach.

NSL-KDD dataset is the subset of KDDCUP99 suggested and derived by Canadian Institute of Cyber Security("NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB", 2017) based on the research by Travallaee. This condensed dataset has removed the limitation caused by the redundant and duplicate records, an unreasonable number of data in testing and training set and furthermore has introduced a new attribute based on the difficulty of detection. Network attacks which are harder to detect are ranked higher than those that are easier. This new attribute will be exploited to generate robust ruleset.

The bar graph below compares the condensed dataset derived from the KDDCup99 dataset. The new dataset has reduced the training set by 78.05% and testing set by 75.15% without omitting any information. (Tavallaee, et al., 2009). The distinct records in the graph shows the size of the reduced dataset compared to the original records in the KDDCup99 training and testing sets.



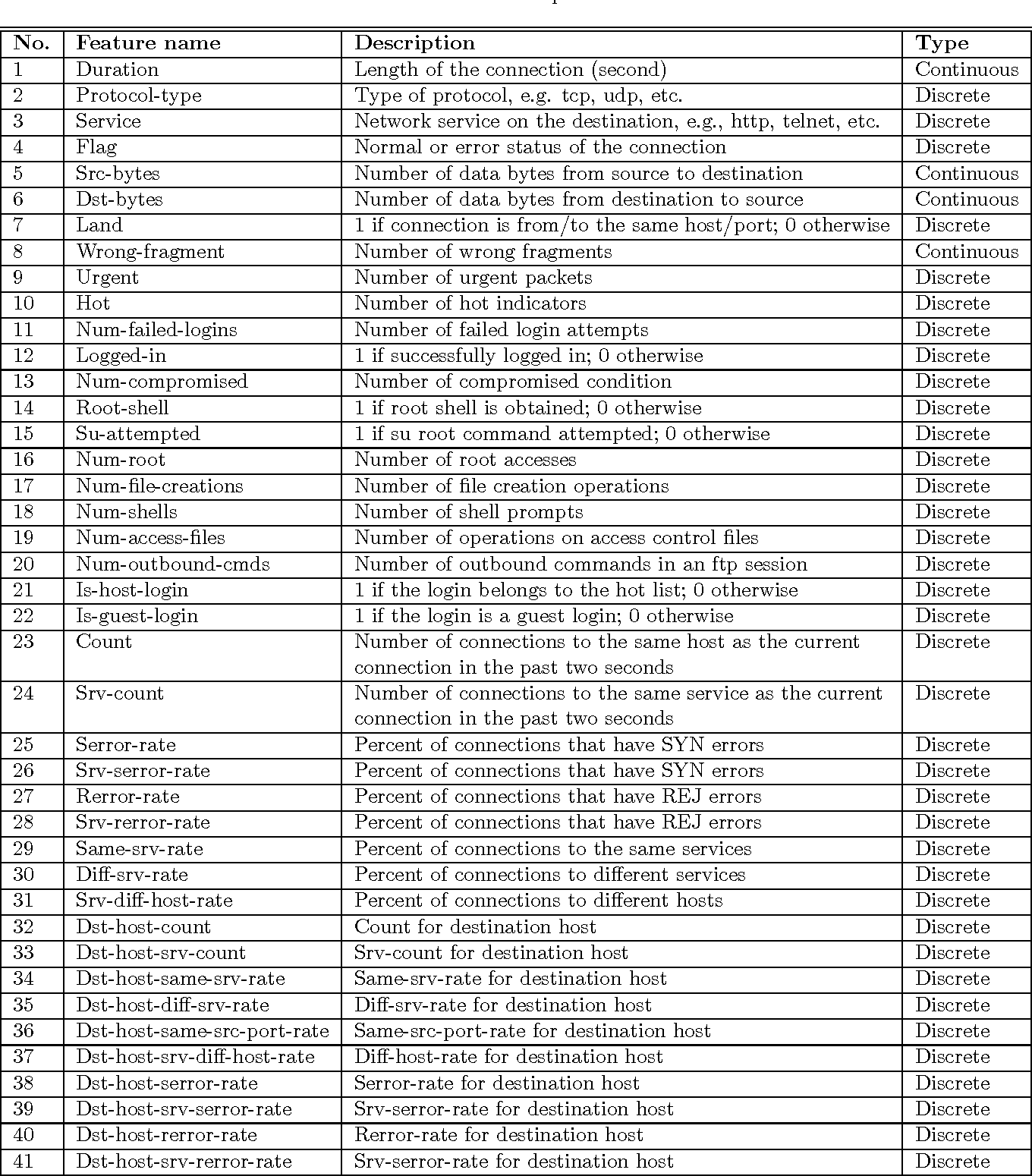
*Graph 1: Statistics of Redundant Records in the KDD train set*



*Graph 2: Statistics of Redundant Records in the KDD test set*

**Network data and their representation.**

The dataset consists of 41 features. The features are shown in the figure below



*Fig 3: Network Features in a training Dataset (Aghdam, M.H., & Kabiri, P.,2016)*

The features above consists of basic network features, traffic features and content features. The basic features are the information contained in the network packet such as protocol type, flags, source bytes etc. The traffic features are computed in a window of time interval and includes features with same host and same service characteristics. The content features are useful to detect remote to local (R2L) and user to root (U2R) attacks. These features includes information from the contents of the packets such as number of login attempts.

**Feature selection**

###### Not all the features in the dataset are significant enough to detect intrusions. Apart from that, the dimension of features is too large to train the system. Each feature must have a binary representation. If the length of the binary string is too long, training the systems will take an absurd amount of time and testing the dataset and implementing the rules will cause system overhead. Selection of features is done using different statistical methods. This section discusses methods of feature selection.

###### ***Information Gain.***

Information gain is very useful measure to determine how much information a feature gives about a certain class. A feature which classifies the class perfectly gives maximal information whereas unrelated information gives no information at all. (*Decision Trees*, n.d.). Information gain is a very efficient way to minimize the expected number of tests required to classify a data.

The expected information needed to classify a tuple in D is given by. (Han, J., Kamber, M., & Pei, J., 2011).

Where is the nonzero probability of tuple D belonging in class.A Feature with multiple values {,….} can divide the dataset into v subsets {,,,……}. A single feature might not correspond to a certain class but rather there might be collection of tuples from different classes rather than a single one. The amount of information needed to find an exact classification is given as:

The information gain is now defined as the difference between the original information and required information. i.e.

Gain (A) =-

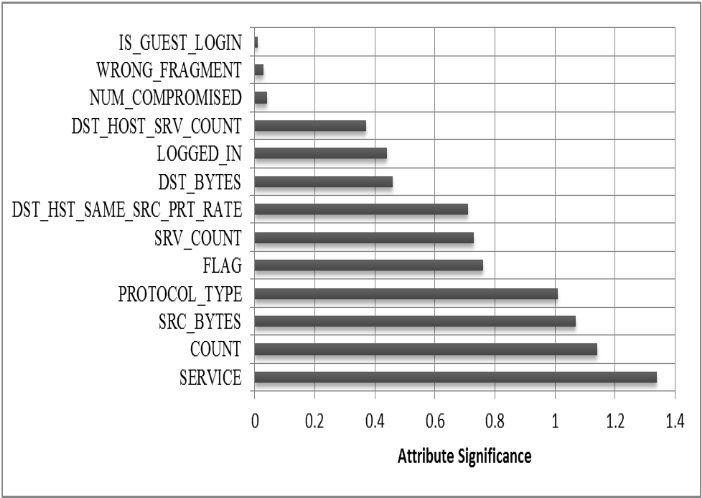
The table below lists the features which has the maximum information gain and are most likely to be a good selection of feature to classify the data. (Abdullah, Abd-alghafar, Salama & Abd-alhafez, 2009).

|  |  |
| --- | --- |
| Classs Label | Relevant Features |
| Normal | 1, 6, 12, 15, 16, 17, 18, 19, 31, 32, 37 |
| Smurf | 4, 25, 26, 29, 30, 33, 34, 35, 38, 39 |
| Neptune | 2, 3, 5, 23, 24, 27, 28, 36, 40, 41 |
| Back | 10, 13 |
| Land | 7 |
| Tear drop | 8 |
| ftp\_write | 9 |
| Guess\_pwd | 11 |
| Buffer\_overflow | 14 |
| Warezclient | 22 |

*Table 4: Classes of attacks with most relevant feature.*

***Attribute Significance***

Attribute importance function is a supervised learning technique which ranks the attributes that are most important in predicting the target attribute. A result of applying the attribute importance function in Oracle Data Miner in the KDD99 dataset shows the ranks of features based on their importance in classifying the data. (Eldos, Taisir & Siddiqui, Mohammad Khubeb & Kanan, 2012)



*Fig4: Attribute significance of features in KDDCup99*

Attribute selection using Information gain is biased towards attributes that have large no of values. This might not always be an ideal solution. We will be using the attributes based on their significance from the table above to train our system.

##### Feature Encoding

As mentioned in the chapter above (canonical genetic algorithms), each feature needs to be encoded into a binary representation. A statistical analysis of the number of bits required to encode the features derived from the attribute significance is defined below.

OpenMl is a free machine learning platform accessible to all users. Datasets can be uploaded and studied in the platform for free. (Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo, 2013). Analysis of the KDDCup99 Dataset was performed using the platform to generate a part of these statistics. Python programming was used to find the unique values in the dataset NSL KDD and will be used to condense the feature encoding.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Min Value | Maximum Value | Unique Values | No of bits required |
| src\_bytes | 0 | 693375640 | 3602 | 30 |
| dst\_bytes | 0 | 5155468 | 10402 | 23 |
| srv\_count | 0 | 511 | 512 | 9 |
| count | 0 | 511 | 512 | 9 |
| dst\_host\_srv\_count | 0 | 255 | 256 | 8 |
| dst\_hst\_same\_src\_prt\_rate | 0 | 1 | 2 | 1 |
| logged\_in | 0 | 1 | 2 | 1 |
| wrong\_fragment | 0 | 3 | 3 | 2 |
| num\_compromised | 0 | 884 | 96 | 10 |
| is\_guest\_login | 0 | 1 | 2 | 1 |
| protocol\_type | - | - | 3 | 2 |
| flag | - | - | 11 | 4 |
| service | - | - | 70 | 7 |
| Total Number of Bits Required | | | | 107 |

*Table 5: Statistical analysis of feature encoding in NSL-KDD dataset*

In order to represent the maximum values of all the features, a total of 107 bits is required to encode all of the selected attributes. This is a really large number considering the efficient machine we are going to propose. In order to reduce the bits needed to encode the features, we created an index dynamically for all the unique values present in the training and testing dataset. We used the index of the unique values to encode the attributes. Using indexing we reduced the number of bits required from 107 to 77 bits. As 77 bits are still computationally lengthy encoding, we will be further selecting the features to generate a more robust model for rule generation.

**Performance Evaluation of the system**

The evaluation of the proposed system uses two standard metrices of detection rate and false positive rate. These metrices are calculated using True-positive, True-Negative, False-Positive and False-Negative measures of the ruleset.

True Positive (TP) occurs when an attack is successfully predicted as an attack and if predicted normal it is termed as False Negative(FN).

False Positive (FP) is when a normal connection is labelled as an attack and False Negative(FN) occurs when normal connection is predicted as normal.

The Detection Rate(DR) and False Positive Rate(FPR) are calculated as follows:

The proposed model and all the other IDS system are measured using these two measures. A high detection rate and low false positive rate is considered as an ideal solution. To aim for the ideal solution a performance evaluation of the generated ruleset and the calculation of fitness of each rule is vital while training the system.

A set of good rules is better than a single best rule. (Sinclair, Pierce, and Matzner 1999) Performance Evaluation helps to find the optimal set of cooperating rules. A geometric mean of true positive rate and a true negative rate is used to calculate the performance of the ruleset. Such a measure tries to maximize the accuracy of on each of the two classes while keeping their accuracy balanced.(Barandela, Sánchez, Garcı́a & Rangel, 2003).

Performance Evaluation:

A performance evaluation is calculated at each generation of the algorithm. The population with the maximum performance will be selected as the ruleset for the IDS system.

**Fitness Function**

Fitness functions evaluates the goodness of the rule. If a rule detects an attack successfully and differentiate it with normal connection, it has a better fitness value. The fitness value is one of the major optimization criteria of the genetic algorithm. The proposed system uses a *support and confidence* framework as the fitness function similar to an approach proposed by Wei Li (Wei Li, 2004). The fitness value of each individual is defined based on the outcome of comparison between a rule and testing data.

As proposed in the feature encoding we have 77 bits to define a rule. The 77 bits of that the algorithm produces as a rule is compared with the encoded features of the ruleset. The outcome is calculated as

If the bits are same it is considered as match and represented by 1 while a mismatch is 0. The weight defines the significance of the feature. If a feature with more significance has more weight. The weight of the features are calculated using the attribute significance measure defined in the chapter above (*Figure 3*). All the bits representing a feature will have the same weight as the feature.

A suspicion level is defined within the algorithm which is a threshold which determines if a classification of the data needs to be performed or not.

For e.g.: If the outcome of an individual in the rule is over the threshold or suspicion level then it is allowed to predict the class of the data. If the predicted class matches the rule it is rewarded but if mismatch occurs it is penalized. This method defines the fitness of the individual.

The penalty is calculated as follows:

*penalty*=

The fitness of the individual is calculated as

Fitness=1-*penalty*

The rank in the equation above is the difficulty level of the network data. If a network data is easy to identify it will have a higher rank and the penalty will be more and vice versa. The rank of each network data is generated using the difficulty label in the NSL KDD dataset as discussed in the chapter before. The dataset has defined the difficulty level from a range of 0 to 21 where 21 is the easiest to detect while 0 is the hardest. We will be scaling this rank from 1-100 to use in the fitness function. The fitness of each individual is in the range of 0-1.

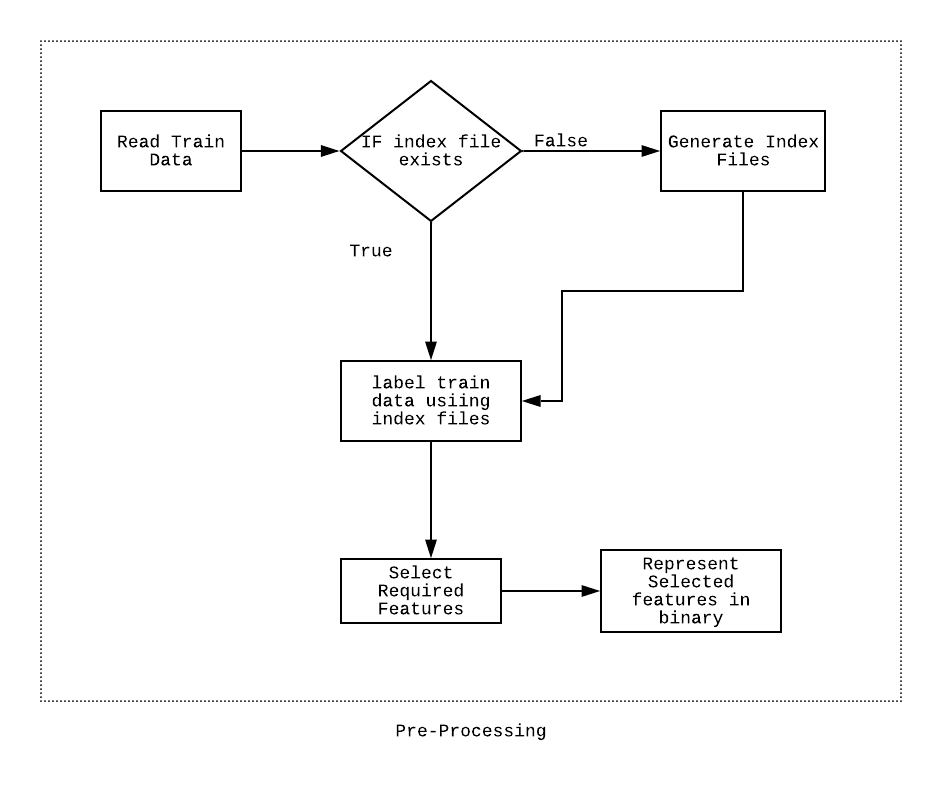
##### A Model of the proposed System

The proposed system is divided into two parts. A preprocessor that processes the input data and converts it into a binary feature which is then used as an input for evolutionary algorithms. The next process then performs evolutionary functions on the encoded dataset to generate rules.

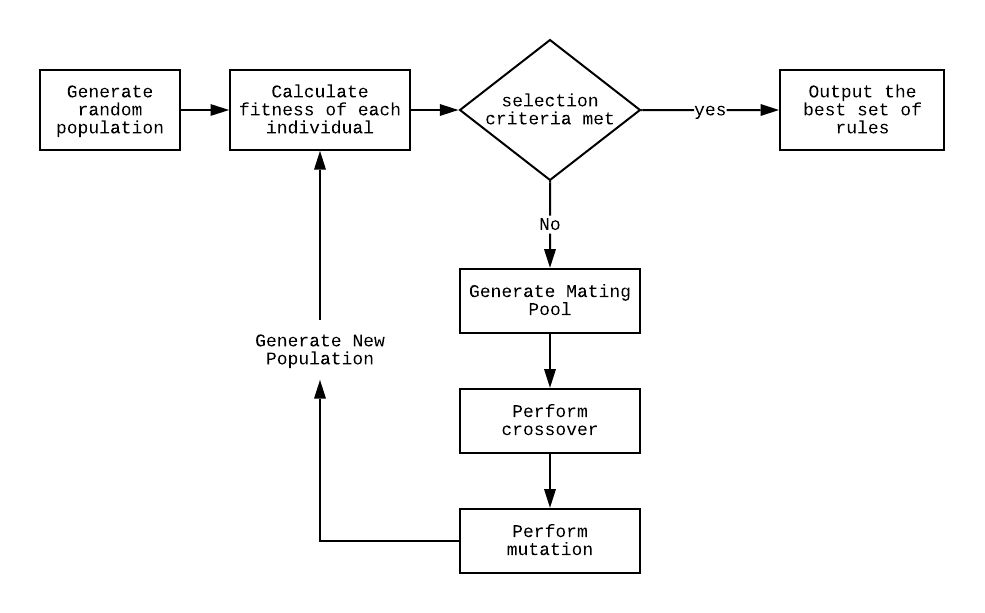
The preprocessor reads data from the training dataset and creates an index file which consists of unique values of each feature. This is done to condense the binary representation of each feature as discussed in the chapter above. The preprocessor uses the index file to assign an index number to each value in the feature set and converts the index to a binary representation. This binary represented data is used to perform evolutionary computing.The encoded data is mostly used to calculate the fitness value of each individual in the evolutionary phase. Each randomly generated individual and individuals from each generation goes through the evaluation function which calculates their fitness by analyzing their performance against the encoded test dataset generated from the preprocessing phase. The fitness function defined in the chapter above is used to calculate the fitness of each individual.

After the fitness value is calculated the population of individuals will be going through the reproduction phase. The best individuals based on their fitness function will be selected for the reproduction process. A mating pool is generated will all the individuals of the population. Two parents are selected from the mating-pool for a crossover process(Fig 2). A tournament selection process (Goldberg & Deb, 1991) selects the two parents for crossover. In this process, few individuals are selected in random from the mating-pool and the one with the better fitness value will be selected as a winner. The two individuals selected as parents will go through the crossover process. Crossover is the mechanism for reproducing. Two children produced from a single point crossover operation go through the mutation process and finally replace the population. The crossover operation has been discussed in the chapter above (canonical genetic algorithm).

A basic flow diagram of both the phases are shown below.



*Fig 5: A preprocessor which encodes the training dataset.*



*Fig 6: A genetic approach to rule generation*

**Algorithm 1**: Preprocessing the Training Dataset

**Result**: Binary Encoded Network Data

**Initialize**:

Declare empty array: *Encoded\_Data*

Read Network Audit Data(NSL KDD Train Set)

For *every record in the train set*:

Initialize empty array *temp\_data*

If (*index file exists*):

Read Index files as *index*

Else:

Initialize empty *index*

For *each feature in the data*:

If value of feature exists in *index*

append index of value into *temp\_data*

else:

update the value in *index*

append the new index of the value in *temp\_data*

Append *temp\_data* into *Encoded\_Data*

For *each record in the Encoded\_Data*:

Select the *Required Features* from the available 41 features.

Convert all the values in the record to binary

Add padding values to make fixed length binary representation for each feature

Output File: *Encoded\_Data*

**Algorithm 2**: Genetic Algorithm For Rule Generation

**Result**: An Efficient Rule Set

**Initialize**:

Read Training Data from output of Algorithm 1 as *training data*

Population size=200

Number of Generation=500

Probability of Mutation=0.01

Generate new population with random set of 200 individuals.

Calculate Fitness()

Current Generation=0

Best Population=[]

While *Current Generation < Number of Generation*:

Create Mating Pool()

For *number of individual in population*:

Select Parent1 and Parent2 from mating pool using Tournament Selection.

Generate Child1,Child2 from Crossover(Parent2, Parent2)

Perform Mutation In Child1 and Child2

Replace Individuals in population with Parent1,Parent2,Child1 and Child2

If *performance of Current Population > performance of Best Population:*

Best Population= Current Population

Increase Current Generation by 1

Fitness():

For *each individual in population*:  
 fitness=0

For *each data in trainset*:

calculate

Penalty=0

If *mismatch:* calculate

*penalty*=

fitness=1-penalty

**Future deployment of the model**

In this paper, we discussed a methodology of applying a genetic algorithm to network intrusion detection. The proposed system is the automation of the rule generation process.

Future work for this study includes the testing of the model using the available dataset. The model will be tested for multiple instances in the various environment to generate an optimized set of rules. Various specification of parameters is purposely vague in the proposal because of the absence of experimental data. The weights of the features, the suspicion level, the number of generation or the size of the population will be decided through extensive tests and statistical analysis.

The output of this model will be a very fit set of rules which can successfully identify attacks and classify normal connection. A rule set for different types of attacks can be generated through this process and such a ruleset can be integrated into “nprobe” to analyze the network data and identify attacks. nProbe is an efficient tool which collects the data from a network, analyses it and dumps into a database. A plugin for nProbe can be developed to integrate the rule generated by our model to analyze the data in the network. The plugin can load the rules from the configuration files and run the network data through each of the rules in the ruleset. If a network data matches a rule an alert can be output to stdout(standard output) which is monitored by the network administrator.

**Conclusion**

This research demonstrates a novel approach to the structure and usage of a rule-based genetic algorithm. Various optimization techniques have been used by different researchers to overcome numerous limitation of the genetic approach and the available dataset. The proposed method is an attempt to collectively address a number of issues related to this domain. The study critically analyzed the predominant dataset and proposed the use of a better subset. Moreover, the model exploits the new attribute “difficulty level” introduced in the NSL KDD dataset to develop a set of rules which will successfully address most of the attack instances. Previous works done by other researches have shown a good detection rate but since the training dataset consists of redundant and duplicate records, the statistics have been misleading and various attacks present in the dataset which are not dominant have been ignored. The study addresses this issue and aims to generate a set of rules which will homogenously detect all the attacks, increasing the detection rate.

## References

Abdullah, B., Abd-alghafar, I., Salama, G., & Abd-alhafez, A. (2009). Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System. *International Conference On Aerospace Sciences And Aviation Technology*, *13*(AEROSPACE SCIENCES), 1-17. doi: 10.21608/asat.2009.23490

*Aghdam, M.H., & Kabiri, P. (2016). Feature Selection for Intrusion Detection System Using Ant Colony Optimization. I. J. Network Security, 18, 420-432.*

Balakrishnan, S., K, V., & A, K. (2014). Intrusion Detection System Using Feature Selection and Classification Technique. *International Journal Of Computer Science And Application*, *3*(4), 145. doi: 10.14355/ijcsa.2014.0304.02

Barandela, R., Sánchez, J., Garcı́a, V., & Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition*, *36*(3), 849-851. doi: 10.1016/s0031-3203(02)00257-1

Bronte, R., Shahriar, H., & Haddad, H. (2016). A Signature-Based Intrusion Detection System for Web Applications based on Genetic Algorithm. *Proceedings Of The 9Th International Conference On Security Of Information And Networks - SIN '16*. doi: 10.1145/2947626.2951964

Chakrabarty, B., Chanda, O., & Saiful, M. (2017). Anomaly based Intrusion Detection System using Genetic Algorithm and K-Centroid Clustering. *International Journal of Computer Applications*, *163*(11), 13-17. doi: 10.5120/ijca2017913762

Chittur, Adhitya. (2001). Model Generation for an Intrusion Detection System Using Genetic Algorithms.

Danane, Y., & Parvat, T. (2015). Intrusion detection system using fuzzy genetic algorithm. *2015 International Conference on Pervasive Computing (ICPC)*. doi: 10.1109/pervasive.2015.7086963

*Decision Trees*. [Ebook]. Retrieved from https://www3.nd.edu/~rjohns15/cse40647.sp14/www/content/lectures/23%20-%20Decision%20Trees%202.pdf

Denning, D. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, *SE-13*(2), 222-232. doi: 10.1109/tse.1987.232894

Desale, K., & Ade, R. (2015). Genetic algorithm based feature selection approach for effective intrusion detection system. *2015 International Conference on Computer Communication and Informatics (ICCCI)*. doi: 10.1109/iccci.2015.7218109

Diaz-Gomez, P.A., & Hougen, D.F. (2005). Improved Off-Line Intrusion Detection Using a Genetic Algorithm. ICEIS.

Eiben, A., Smith, J.E. (2003). Introduction to evolutionary computing. *Assembly Automation,* *24*(3), 324-324. doi:10.1108/aa.2004.24.3.324.1.

Eldos, Taisir & Siddiqui, Mohammad Khubeb & Kanan, Aws. (2012). ON THE KDD'99 DATASET: STATISTICAL ANALYSIS FOR FEATURE SELECTION. Journal of Data Mining and Knowledge Discovery.

Forrest, S., Perelson, A., Allen, L., & Cherukuri, R. (2002). Self-nonself discrimination in a computer. Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy.

Gauthama Raman, M., Somu, N., Kirthivasan, K., Liscano, R., & Shankar Sriram, V. (2017). An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*, *134*, 1-12. doi: 10.1016/j.knosys.2017.07.005

Goldberg, D., & Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations Of Genetic Algorithms*, 69-93. doi: 10.1016/b978-0-08-050684-5.50008-2

Han, J., Kamber, M., & Pei, J. (2011). *Data mining : Concepts and techniques* (3rd ed., Morgan kaufmann series in data management systems) [3rd ed.]. Burlington: Elsevier Science. (2011). Retrieved June 4, 2019, from INSERT-MISSING-DATABASE-NAME.

Ijaz, S., Hashmi, F., Asghar, S., & Alam, M. (2017). Vector Based Genetic Algorithm to optimize predictive analysis in network security. *Applied Intelligence*. doi: 10.1007/s10489-017-1026-9

Jabez, J., & Muthukumar, B. (2015). Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. Procedia Computer Science, 48, 338-346. doi: 10.1016/j.procs.2015.04.191

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo.(2013) OpenML: networked science in machine learning. SIGKDD Explorations 15(2), pp 49-60,

Jones,Anita K. and Robert S.Sielken. (2000). Computer System Intrusion Detection: A survey. Technical Report Department of Computer Science, University of Virginia, Charlottesville Virginia

KDD Cup 1999 Data. (1999). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Li, Wei. (2004). Using genetic algorithm for network intrusion detection.

Miller, B., & Shaw, M. (1995). Genetic algorithms with dynamic niche sharing for multimodal function optimization. Proceedings of IEEE International Conference on Evolutionary Computation. doi: 10.1109/icec.1996.542701

NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (2017). Retrieved from https://www.unb.ca/cic/datasets/nsl.html

Pohlheim, H. (2003). “Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms.” Genetic and Evolutionary Algorithm Toolbox. Hartmut Pohlheim. URL: http://www.geatbx.com/docu/algindex.html.

Rastegari, S. (2015). *Intelligent network intrusion detection using an evolutionary computation approach*(Doctoral dissertation, 2015). Edith Cowan University, Edith Cowan University, Research Online, Perth, Western Australia, Perth, Western Australia.

S. Selvakani and R.S. Rajesh (2007), “Genetic Algorithm for Framing Rules for Intrusion Detection” IJCSNS International Journal of Computer Science and Network Security.

Sujatha, K., Dharmar, V., & Bhuvaneswaran, R. (2012). Design of genetic algorithm based IDS for MANET. 2012 International Conference on Recent Trends in Information Technology. doi: 10.1109/icrtit.2012.6206796

Tao, P., Sun, Z., & Sun, Z. (2018). An Improved Intrusion Detection Algorithm Based on GA and SVM. *IEEE Access*, *6*, 13624-13631. doi: 10.1109/access.2018.2810198

Tavallaee, M., Bagheri, E., lu, W., & Ghorbani, A. (2009). A Detailed Analysis of the KDD. *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*. Retrieved from http://dl.acm.org/citation.cfm?id=1736481.1736489

Thanuja, R., & Umamakeswari, A. (2018). Black hole detection using evolutionary algorithm for IDS/IPS in MANETs. *Cluster Computing*. doi: 10.1007/s10586-018-2006-5

The Nature of Code. (2018). Retrieved from https://natureofcode.com/book/

Vanschoren, J. OpenML. Retrieved from https://www.openml.org/d/1110

Vijayanand, R., Devaraj, D., & Kannapiran, B. (2018). Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. Computers & Security, 77, 304-314. doi: 10.1016/j.cose.2018.04.010

Whitley, D. (1994). A genetic algorithm tutorial. Statistics and Computing, 4(2). doi: 10.1007/bf00175354