Rough Tree Classifier for Network Intrusion Detection with GA-Based Feature Selection

Developed by

Rubbiya Akram Ali

[349-FBAS/MSCS/F07]

Supervised by

Assistant Professor Qaisar Javed

Department of Computer Science

Faculty of Applied Sciences

International Islamic University, Islamabad

(2009)

Allah's Name I Begin With, the Most Compassionate, the Most Merciful

International Islamic University, Islamabad

Dated: September 12, 2009

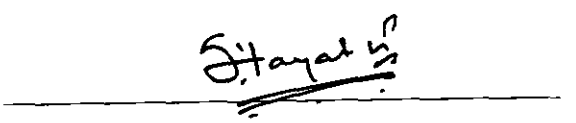
Final Approval

It is certified that we have read the project report, titled "**Rough Tree Classifier for Network Intrusion Detection with GA-Based feature selection**" submitted by **Rubbiya Akram Ali.** It is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad, for the Degree **MS in Computer Science**.

Committee

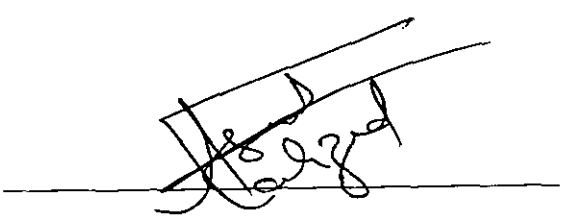
External Examiner

Prof Dr. Sikandar Hayat Khiyal,
Chairperson,
Department of Computer Science,
Fatima Jinnah Women University,
Rawalpindi



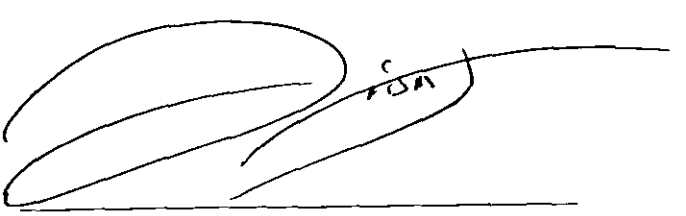
Internal Examiner

Muhammad Shahzad Ashraf,
Lecturer,
Department of Computer Science,
International Islamic University,
Islamabad



Supervisor

Qaisar Javed,
Assistant Professor,
Department of Computer Science,
International Islamic University,
Islamabad



Dedication

Dedicated to the almighty **ALLAH** and **The Holy Prophet Muhammad** (Allah's grace and peace be upon him) lord of the world and the thereafter. I offer my humblest thanks to Him, who made us aware of our creator and guided us to the track, which leads to the success, who is a symbol of love and affection for all the creatures of Allah.

To,

My Loving Parents
who always Encouraged me
And provide facilities to achieve my goals,
And without whom, I am nothing...

A dissertation Submitted To

Department of Computer Science,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of *Master of Science in Computer Science*

Declaration

I hereby declare that this Thesis **"Rough Tree Classifier for Network Intrusion Detection with GA-Based Feature Selection"** neither as a whole nor as a part has been copied out from any source. It is further declared that I have done this research with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of my teachers especially my supervisor **Assistant Professor Qaiser Javaid, IIUI**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from any of the training institute or educational institutions, I shall stand by the consequences.

[Rubbiya Akram Ali]

[Registration # 349-FBAS/MSCS/F07]

Acknowledgements

First of all I pay my humble thanks to **Almighty Allah** who knows all the things hidden or evident in this universe, even the things which pass through our hearts, who created us and gave us the courage to complete this project. It is said in the Holly Quran:

"Does man think that he will be left uncontrolled, (without purpose)? Was he not once a drop of ejected semen? Then he became a clot, so He created and fashioned him and made him into two sexes, male and female. Is He who does this not able to bring the dead to life?" [Surah alQiyama: 36-40]

Without His help and blessings, we were unable to complete the project.

I also offer my humblest thanks to **The Holy Prophet Muhammad** (Allah's grace and peace be upon him), who made us aware of our creator and guided us to the track, which leads to the success, who is a symbol of love and affection for all the creatures of Allah.

I also express my cordial and humble thanks to my supervisor **Assistant Professot Qaisar Javed** for his untiring help and cooperation in completing this project.

Special Thanks to **Mr. Ahmed Murtaza Mehdi,** who always spare a lot of time and guided me throughout the research. Without his continuous Technical help it was impossible for me to enter in the heavens of *"Machine Learning"*.

I cordially regard the inspiration, prays, encouragement and financial support of my loving and affectionate parents and family for their motivation in every aspect of my study enabling me to complete this project.

[Rubbiya Akram Ali]

[Registration # 349-FBAS/MSCS/F07]

Project in Brief

Project Title:	Rough Tree Classifier for Network Intrusion Detection with GA-Based Feature Selection
Objective:	Improving accuracy by falling unrelated and possibly unneeded features
Undertaken By:	Rubbiya Akram Ali
Supervised By:	Assistant Prof Mr. Qaiser Javaid
Start Date:	19-02-2009
Completion Date:	31-08-2009
Technologies Used:	• JAVA (Eclipse) • Matlab 7.0 • Graphviz 2.24
Documentation Tools	Microsoft Office Tools
System Used:	Pentium® III
Operating Systems Used:	Microsoft® Windows® XP Professional

Abstract

Due to the gigantic increase in the information sharing between different groups of people, Computer Networking plays an important role for their connection. The security in the information processing is the major issue that is always given priority in different network topologies and Computational Intelligence algorithms have been proved to be best suited for detecting the different attacks on the network and emerged as an imperative procedures for enhancing the security. In this study I want to present an intrusion detection system that could increase the detection and decrease the false alarm rate in network intrusion detection. A number of techniques have been presented to deal with it. But my proposed technique is a hybrid network intrusion detection system that could counter better performance by selecting only relevant features. The proposed system is based on the hypothesis that not every feature of the training data may be relevant to the detection task and, in the worse case, irrelevant features may introduce noise and redundancy into the design of classifiers and choosing a good subset of features will be critical to improve the performance of classifiers.

TABLE OF CONTENTS

Chapter 1
Introduction

CHAPTER 1 Introduction

Information today can either make or break companies and it should be protected. Having, IDS that can distinguish between normal use and an intrusion, without the use of a network analyst to police the network, will help protect that information without fear of it being view by others.

1.1 Motivation and Challenges

In modern society, network-based computer systems are playing vital roles in today's society; such systems are mostly targeted by our enemies and criminals. Therefore, the need of the time is to find out the best ways to defend our systems and more provide them more secure environment. The reason is that the sophistication of attacks has increased and the level of difficulty to intrude on a system has decreased. This is due to all the new technology and the resources available to those wanting to learn how to intrude on a system. This is where intrusion detection systems play a big role. Having an intrusion detection system will help protect that private information without the constant fear of it being view by others.

The safety of a computer system is negotiated when an interruption takes place. Intrusion detection therefore contains these necessary elements:

- Resources of the target system must be protected.
- Models which classify the "normal" or "lawful" behavior of the activities involving these resources;
- Method that compare the observed behavior with the standard models. And after that the connections or activities that are not considered as "normal" are flagged as "intrusive".

With the increasing popularity and usage of machine learning techniques to intrusion detection, a problem, of how to choose the features (attributes) of input training data on which learning can take place, still can be improved by further development. Many approaches like decision trees, Genetic Algorithm and Genetic Programming, naive Bayes, kNN and neural networks have been used for security but they all proved to be unsatisfactory from different context.

1.2 Background:

1.2.1 Intrusion:

An unauthorized access to the resources of the computer is called an intrusion to a computer. Intrusion can be said a concatenated series of such activities that are dangerous

to the security of IT resources from unauthorized access to a specific computer or address domain. [2]

1.2.1.1 Intrusion Detection System:

Intrusion Detection System is a mechanism procedure which provides safety and checks for illegitimate users who causes dangerous activities which results as intrusions to computer. IDS must differentiate between legitimate and illegitimate activity to the computer resources in order to distinguish between intrusions and non intrusive activities. In recent years, dramatically increase the amount of data (text; images; audio; etc.) that available electronically on the Internet. For the security of data over the network and internet, many techniques have been introduced. The number of computer attacks has increased exponentially in the past few years. In a report presented by the Computer Emergency Response Team/Coordination Center (CERT/CC), the graph of the growth rate of cyber incidents was noticed. The graph shows the increasing trend of attacks from 1990 to 2003. [2]

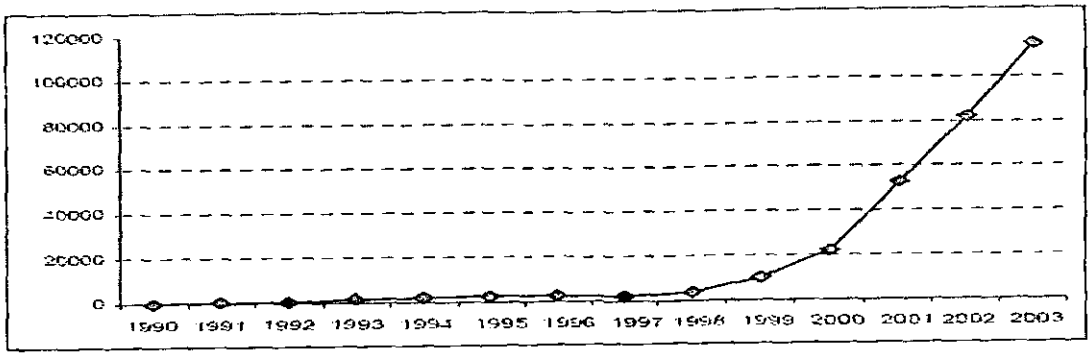


Fig1: The growth rate of Cyber incidents [2].

So, Intrusion detection system (IDS) congregates and examines information from a variety of areas in a computer or a network to recognize probable protection breaches, which include both intrusions and mistreated. IDS are software systems designed to recognize and avoid the misuse of computer networks and systems.

1.2.2 Types of Intrusion Detection Systems:

Intrusion detection methodologies can be further divided into the following types

1. Network based Intrusion Detection
2. Host based Intrusion Detection
3. Hybrid Intrusion Detection

1.2.2.1 Network Intrusion Detection (NID)

when some information is passed on the wire between hosts, and some intrusive behavior detected, then such detection is called Network Intrusion detection. This intrusive behavior is also known as "packet-sniffers,". When the packets are roaming between various communication channels and protocols, Network Intrusion Detection devices used to capture them. When the packet start wandering in the network, it is the duty of NID device to associate it with marked database consisting of identified attacks and non-required packets,

Some NID devices will simply associate the packet to a mark database consisting of identified attacks and non-required packet "fingerprints", whereas others will checks for irregular packet activity that might denote malicious behavior. In either case, network intrusion detection should be considered largely as a boundary defense. NID Systems monitor multiple hosts by analyzing network traffic. Snort is one of the examples of NIS systems.

1.2.2.2 Host Intrusion Detection (HID)

The start of Host-based intrusion detection was in the early 1980s before. In that age, networks were as widespread, complex and interrelated as they are today. The duty of HID systems was common exercise to analyze audit logs for doubtful actions. Such systems provided such an environment, where intrusions were adequately rare. When the analysis proved enough, it automatically makes a system generic to avoid future attacks. (HIDS) consists of an agent on a host which identifies intrusions by observing the calls, application logs etc.

1.2.2.3 Hybrid Intrusion Detection

Hybrid intrusion detection systems manage alert notification from both network intrusion detection systems and host-based intrusion detection devices. It acts like a central intrusion detection management and provides the coherent supplement between NID and HID.

Hybrid Intrusion Detection is a combination of two or more techniques/ approaches. Network information is combined with the Host agent data which results a very comprehensive view of the network. An example of Hybrid IDS is Prelude.

1.2.3 Why Data Mining is needed?

Prediction and Description are the two high-level primary goals of data mining. Predictions keep using several variables or fields in the already observed records to forecast unknown, unidentified or future values of other variables under concern. Description focuses on judging human-interpretable patterns presentation the data. The objectives of prediction and explanation can be achieved using a variety of scrupulous data-mining methods like,

1.2.3.1 Classification

Classification techniques intend to recognize the characteristics that signify the group to which each instance belongs. This prototype can be used both to understand the obtainable data and to guess how new instances will behave. E.g:, you may want to guess whether individuals can be classified as expected to respond to a straight mail solicitation, susceptible to toggling over to a rival long distance phone service, or a good nominee for a surgical technique.

1.2.3.2 Clustering

Clustering techniques address segmentation questions. These approaches allocate records with a big number of attributes into a comparatively small set of collections or "segments." This allocation process is performed instinctively by clustering algorithms that identify the unique characteristics of the dataset and then divide the x-dimensional space defined by the dataset attributes along natural restrictions. There is no need to identify the groupings preferred or the attributes that should be used to section the dataset. Clustering is normally one of the first steps in data mining analysis. It distinguishes groups of associated records that can be used as a preliminary point for exploring further relations.

1.2.3.3 Association Rule

If the goal of a data-mining question is to discover the relationship between the different variables in a dataset, association rule is used. The famous Aprior algorithm is one of the most famous techniques used for finding the association of different variables.

1.2.3.4 Regression

Whenever there is a need to use the existing values for guessing the future values, regression techniques can be used. Mostly linear regression techniques are used when the relationship between the input and output dataset is known to be linear.

1.2.3.5 Summarization

Summarization engages techniques for ruling a dense description for a subset of data. A simple example would be charting the averages and st. deviations for all subjects. More complicated methods engage the origin of summary rules, multivariate prophecy techniques etc.

1.3 Data mining approaches toward intrusion detection

Intrusion Detection uses Data mining approaches as new methods. I can define data mining as the semi repeated detection of patterns, relations, transformations, irregularities, policies, and statistically important organization and events in data. To describe the system's behavior, knowledge is extracted in the form of models from data through data mining. Such knowledge may not be seen easily with the naked eye. To complete such purpose, many different types of data mining algorithms like classification, regression, clustering, association rule abduction, deviation analysis, sequence analysis etc are working. These algorithms are used according to their nature.

Due to the benefit of determining supportive knowledge which illustrates a user's or program's performance from large audit data sets, diversive data mining procedures have been applied to intrusion detection. [1]

1.3.1 The data mining process to build intrusion detection models

1. First Raw (binary) data is processed into ASCII network packet information (or host event data).
2. This processed packet information is in turn sum up into connection records (or host session records). These records contain a number of within-connection features, e.g., service, duration, flag etc.
3. Then Data mining programs are applied to the connection records which compute the recurrent patterns i.e. association rules and recurrent episodes

4. These frequent patterns are analyzed to build added features for the association records.
5. Classification algorithms are then used to discover the detection model.

This process continues iteratively e.g., poor performance of the classification models often signifies that additional data recognition and feature construction is needed. [1]

1.4 What is Machine Learning?

ML usually refers to the changes in the system that performs tasks associated with artificial intelligence (AI). Such responsibilities involve identification, analysis, Planning, automaton control, forecast etc. The "changes" might be either enhancements to already performing system or an initial synthesis of new systems. To be slightly more specific, I show the architecture of a typical AI "agent" in the Fig 1. This agent perceives and models its environment and compute appropriate actions, perhaps by anticipating their effects. Changes made to any of these components shown in the figure might count as learning. Distinctive learning mechanisms force to be working depending on which subsystem is being distorted.

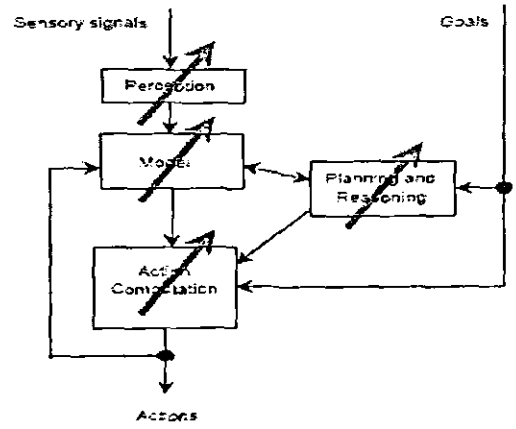


Fig.2 AI Architecture

There are numerous AI techniques which have been consumed to mechanize the intrusion detection process to diminish human interference some of AI techniques are neural networks, KNN's , fuzzy inference systems, evolutionary computation machine learning, etc data mining techniques have been introduced to identify key features or parameters that define intrusions. [15]

1.5 Thesis Motivation and Objectives

The main focus of my thesis is 'How to select the features (attributes) of the input training data on which learning will be obtained'. Everyone knows that from a set of attributes, not every feature of the training data may be informative or relevant to the detection task and Features may contain false correlations, which delay the process of detecting intrusions. So, a blind choice of feature selection unavoidably include many irrelevant features which may introduce noise and redundancy into the design of classifiers, and will definitely do harm to classification performance, can increase computation time, and can impact the accuracy of IDS. [3]

Feature selection has been the center of attention in statistical pattern detection, machine learning, and data mining. Choose a best subset of features has confirmed in both theory and practice effective in attractive learning efficiency, escalating predictive accuracy, and reducing complexity of learned results.

But, most favorable attribute selection requires an exponentially huge search space, where N is the number of attributes. So it may be too expensive and unreasonable.

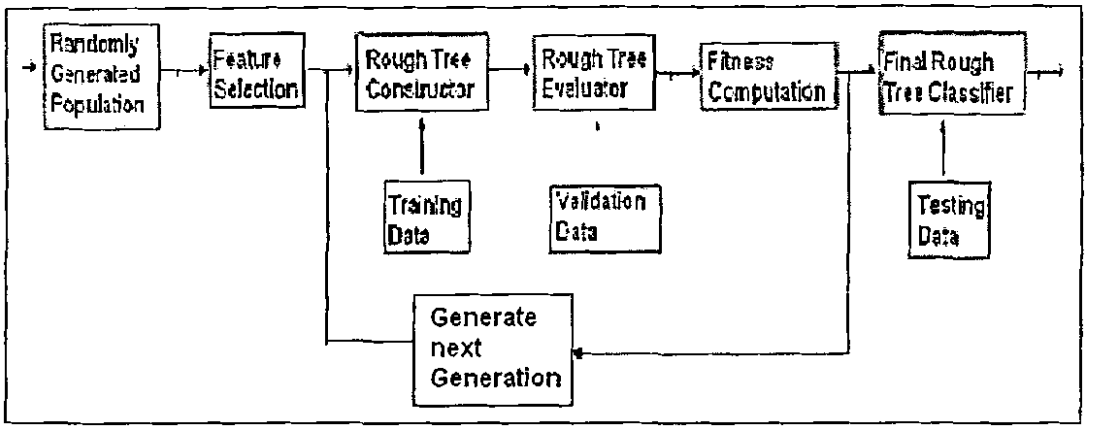
To reduce the number of features, there are many reasons for using feature selection technique which really works.

1. Ignoring or falling unrelated and possibly unneeded features from features dataset really improves accuracy.
2. Fulfilling the common goal of increasing the accuracy of the classifier lessens the measurement costs
3. It also Reduce the complexity and the overall associated computational cost
4. The main advantage of reducing features is that it also reduce the amount of data need for the training
5. The probability will be improved and a solution will both understandable and practical. [7]

1.6 Proposed Approach:

In my proposed algorithm, I will use genetic algorithm to select a subset of input features for rough tree classifiers Main goal is to increase the detection rate and decrease the false alarm rate in network intrusion detection. KDDCUP 99 data set will also be used to train and test the rough tree classifiers. The experiments will be done to show that the resulting rough trees will have better performance than those built with all available features. I will only consider misuse detection.

A Detailed description of this algorithm is presented through the following diagram



1.7 Thesis Outline

In chapter 1, '**Introduction**' of each and every concept related to my thesis is given. In chapter 2, '**Literature Review**' a detailed literature review is given. A period wise description of techniques, 'how they were used?', 'what were their deficiencies?', and 'how they contributed in the field of intrusion detection?' is reviewed. In chapter-3, '**Requirement Analysis**' is done. 'What are the requirements of the proposed application', 'why I selected techniques like GA and Rough Tree', 'what models I will follow'. In next chapter '**Design**', all the designing phase is explained. The purpose of design is to create architecture for the developing implementations. So the pre-requirements of the architecture and implementation are explained. the working of Genetic Algorithm , the working of Rough Tree , the flow of work are described well in this chapter. Then, in chapter 5, '**Implementation**', the flow diagrams, the algorithms, deep discussion about dataset, the environments in which the code is developed and code of main processing are available. In chapter 6, '**Results**' of experiment are prepared. 'how they improved the accuracy' , and 'how the are better then the previous results' are explained graphically and table wise. In last chapter, '**Conclusion**' whole summary of thesis with conclusion is mentioned. I also explained the points to be researched for future work of my thesis.

Chapter 2
Literature Review

2. Literature Review

2.1 Introduction:

Intrusion detection is one of the crucial problems at this time in the field of computer networking. With the passage of time, technologies are changing and internet traffic is exponentially growing, which makes the services provided by the intrusion detection system, unreliable. So, the only need at this time is find out such systems which

- Improve classification performance
- Decrease computation time
- Increase the rate of accuracy of IDS

In past few years a lot of work has been done on the above issues. So many techniques developed and applied to achieve improvement. Many techniques really enhance the previous results and give more reliable output, and many were failed from other aspects. a lot of work contributed a lot in this field.

2.2 Related Research and their limitations:

In 1999, the KDD conference hosted a classifier learning contest [16], their learning objective was to build such a model which predicts main points to differentiate attacks and normal connections. MIT Lincoln Labs provides an intrusion dataset to contestants to train and test their classifiers. Each record of this dataset has 41 features. These attributes lies in the following three categories:

(1) Basic features related to individual TCP connections;

(2) Content features within a connection; which contains all the related information of a connection;

(3) Traffic features calculated using a two-second time window.

Their purpose was classification of network connections by automatically generating rules. To develop such rules which match only the irregular connections, genetic algorithm is used which also checks network traffic.

In [5], presented in 2000, Athanassios and Dimitrios presented a new technique GATree, which is little addition to simple GAs. For this extension especially GAs used to directly develop binary decision trees. They selected Gas to develop accurate as well simple decision trees. As a result, GATree produces a dynamic, small biased, exactness/size based tree optimization. But in this process time burden increased.

In 2002, paper [10] focuses to alert the system Administrator about some possible security violation. But in many cases it results in overhead. So in order to decrease the

overhead and to detect both anomalies and misuses, Neural Networks and Support Vector Machines are separately trained to learn normal behavior and attack patterns. As a result significant deviations were flagged as attacks. After experimentation over DARPA KDD dataset SVM give greater than 99% accuracy then Neural Networks IDS. The training time is also significantly shorter, so its performance was quicker when new attack patterns were discovered. On the other hand, SVMs can support only binary classification which is a severe disadvantage because IDS requires multiple-class identification (e.g., all different types of attacks need to be differentiated).

In [11] Amor et al compares the presentation of Naïve Bayes with Decision Trees techniques. Naïve Bayes is organizeed in two stages: (I) origin node which denotes a assembly class (usual and unusual attacks), and several leaf nodes, each representing feature of a connection. However NB assumes that all features are independent in the context of session class.

Decision Tree Classifier directs to weak learning. Paper concludes that learning and classifying with NB Classifier is 7 times faster then learning and classifying with DT Classifier.

In [12], a study is arranges on Classification Techniques for NIDs. This paper [2005] contains a comparative study between K-mean nearest neighbors classifier, Artificial Neural networks Classifier and Support Vector Machines. Authors conclude that

- K-Means Classifier performed poorly with accuracies not exceeding 18%.
- ANN's performance is better then K-Mean Classifier. ANN and SVM both's results are highly accurate, within similar level of performance.
- But, SVMs are far superior then ANNs as its training time is significantly shorter (100 secs compared to 55 minutes).

In [3], paper [2005] presents genetic algorithm approach which selects a subset of input features for decision tree classifier to increase the detection rate and decrease the false alarm rate. For training and testing the decision tree classifiers KDDCUP 99 dataset is used. This Hybrid approach eliminated unnecessary features and only focus on relevant features, which make hybrid to outperform decision tree algorithm without feature selection and improves the classification process of decision trees. This algorithm takes longer to execute then the standard decision tree, but it is better to be adopted as its non-deterministic process can make better decision trees.

Another paper [13] was presented in 1997. Most of the previous studies were done on small databases. This hybrid approach attempts to consume the advantages of both Classifiers i.e., Decision Tree (Segmentation) and Naïve Bayes (evidence accumulation from multiple attributes). Each segment of the data, represented by a leaf, is described through a Naïve-Bayes classifier. So, NBTree induces highly accurate classifiers in practice and appears to be a feasible approach to inducing classifiers, In practice, NBTrees are shown to scale to large databases and, in general, do better then Decision Trees and NBCs alone.

RoughTree (RT) [14] is newest Classifier, with Naïve Bayes and Rough Sets mixed in Decision Tree nodes. The outcome of RT is a tree-like representation and each leaf is substituted by a Naive-Bayesian decision algorithm. RT gets rid of the feature dependences in its nodes and the investigational results show that RT can attain better presentation than Naive Bayesian algorithm.

2.3 Summary:

The main focus of my thesis is 'How to select the features (attributes) of the input training data on which learning will be obtained'. Everyone knows that from a set of attributes, not every feature of the training data may be informative or relevant to the detection task and Features may contain false correlations, which delay the process of detecting intrusions. So, a blind choice of feature selection unavoidably include many irrelevant features which may introduce noise and redundancy into the design of classifiers, and will definitely do harm to classification performance, can increase computation time, and can impact the accuracy of IDS.

2.3.1 *Proposed Solution*

Solution to above stated problem can be achieved by using the Genetic Algorithm with Rough Tree Classifier. This GA-Based Feature Selection algorithm will be based on Wrapper Model. In my proposed algorithm, the Search component will be a GA and the evaluation Component will be a Rough Tree.

2.3.2 GA-Based Feature Selection for Rough Trees:

After reviewing all the literature, I selected two papers [3] and [14].

Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection

Gary Stein, Computer Engineering, University of Central Florida, Orlando, FL 32816-2362, gstein@mail.ucf.edu
Bing Chen, Computer Science, University of Central Florida, Orlando, FL 32816-2362, bchen@cs.ucf.edu
Annie S. Wu, Computer Science, University of Central Florida, Orlando, FL 32816-2362, aswu@cs.ucf.edu
Kien A. Hua, Computer Science, University of Central Florida, Orlando, FL 32816-2362, kienhua@cs.ucf.edu

In [3], GA/ Decision Tree Hybrid were implemented, in which GA was used to select a subset of input features and decision tree was used as classification technique.

Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection

Gary Stein, Computer Engineering, University of Central Florida, Orlando, FL 32816-2362, gstein@mail.ucf.edu
Bing Chen, Computer Science, University of Central Florida, Orlando, FL 32816-2362, bchen@cs.ucf.edu
Annie S. Wu, Computer Science, University of Central Florida, Orlando, FL 32816-2362, aswu@cs.ucf.edu
Kien A. Hua, Computer Science, University of Central Florida, Orlando, FL 32816-2362, kienhua@cs.ucf.edu

In [14]. Authors of this paper developed a semi-naive classifier and named it as **RoughTree.** The purpose of the tree is to deal with the attribute interdependence problem of Naive Bayesian Classifier and alleviate it.

In Rough sets there is a measure which detects the attribute dependence detecting measure. Rough Tree uses this measure and divides the dataset into subspaces according to the selected attributes, which calculate the highest values of attributes by the attribute dependence measure. These values show total contribution of each attribute. This process carry on the same way a decision tree divides until the stopping criterion is fulfilled. Then, the outcome of this process will be represented in the form of tree and each leaf of the tree in the RoughTree is replaced by a Naive-Bayesian classifier. RoughTree get rid of the dependency among the attributes in its leaves and the experimental results show that RoughTree easily can attain enhanced performance and more reliable results than Naive Bayesian classifier.

2.4 Why Rough Tree as Classifier?

I selected RoughTree as Classifier on the following bases:

- SVM Classifier is better then the Neural Networks Classifier. The training time of SVM is significantly shorter, so its performance was quicker when new attack patterns were discovered. On the other hand, SVMs can support only binary classification which is a severe disadvantage because IDS requires multiple-class identification (e.g., all different types of attacks need to be differentiated).(2002)
- So decision Tree Classifier is better then SVM Classifier as SVM is only a binary Classifier.
- Naïve Bayes classifier is competitive and required less training time than the decision tree classifier. Decision Tree Classifier directs to weak learning. Paper concludes that learning and classifying with NB Classifier is 7 times faster then learning and classifying with DT Classifier. So NB is better then DTC. However NB assumes that all features are independent in the context of session class. (2004)
- ANN Classifier is better then K-Mean Classifier. But, SVMs are far superior then ANNs as its training time is significantly shorter (100 secs compared to 55 minutes).(2005)
- Concluding all, I can say that RoughTree Classifier will result better then all the previous classifiers.

So, in this proposed technique, chances of improvement can be seen clearly. In the next chapter, I will explain in detail the problem domain.

3. Requirement Analysis

3.1 Introduction:

Now a days, organizations having large networks are facing problem regarding Intrusion in their networks, which is causing them a big threat. Threats like dos, probe, R2L etc. so the basic problem is to identify these attacks from normal network traffic in a cost effective yet in terms of quality/detection rate perform as good as native systems.

Intrusion detection is one of the crucial problems at this time in the field of computer networking. With the passage of time, technologies are changing and internet traffic is exponentially growing, which makes the services provided by the intrusion detection system, unreliable. So, the only need at this time is find out such systems which

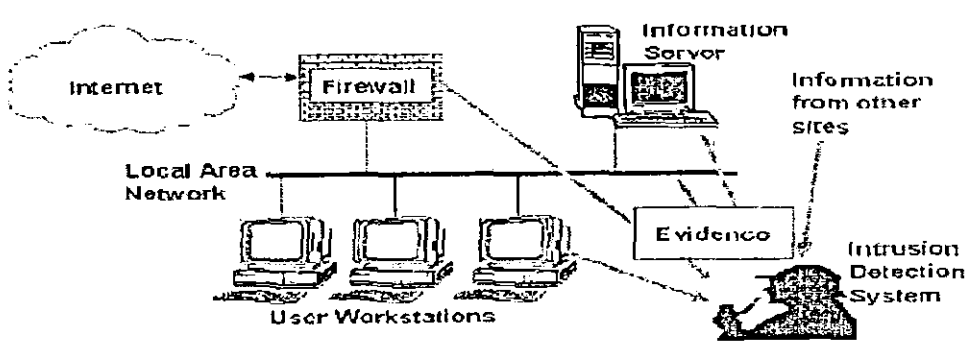
- Improve classification performance
- Decrease computation time
- Increase the rate of accuracy of IDS

Intrusion is actually when the user takes an action that the user was not legally permitted to take place. Intrusion attempt (Anderson, 1980) is defined to be potential possibility of an unauthorized attempt to

- Access information
- Manipulate information, or
- Make a system unreliable or unusable.

It involves determining that an intruder has tried to gain or has gained unauthorized access to the system.

the majority intrusion detection systems effort to detect a reputed intrusion and alert a system administrator.

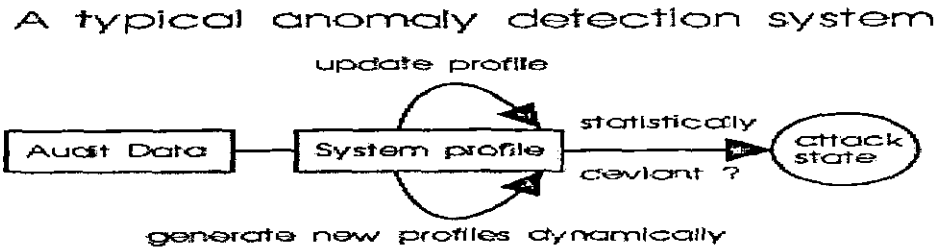


- **Intrusion detection systems monitor network state looking for unauthorized usage, denial of service, and anomalous behavior**
- **Such systems have never been formally evaluated... until now**

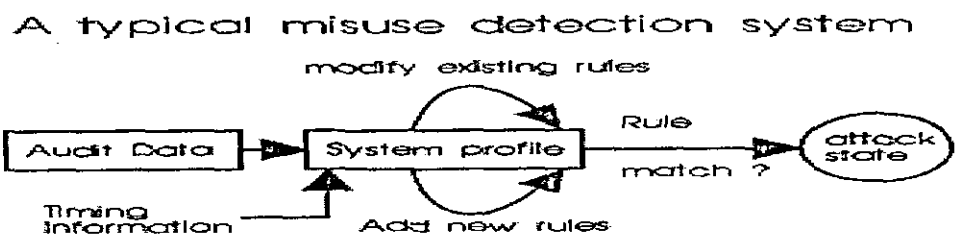
3.1.1 Categories of Intrusion Detection Systems:

There are two main categories of intrusion detection systems:

3.1.1.1 Anomaly detection systems (ADS) seek to identify deviations from normal behavior models which are built from large training data sets. So, detectors build profiles representing normal behavior of users, hosts, or network connections. These outlines are created from sequential data collected over a period of usual process. Then these detectors gather event data and use a range of events to decide when examined activity diverges from the standard.



3.1.1.2 Misuse detection systems (MDS) compare the system user actions with signatures take out from already known attacks; a better match with a high confidence is considered an attack\intrusion. Misuse ID approach concerned with catching intruders who are attempting to break into a system using some known technique.



3.2 Types of Errors

Types of error that may likely to occur can be categorized into

3.2.1 False positive Error occurs when systems identify an action as anomalous when it is a legal action. False positive errors will guide users of the intrusion detection system to pay no notice to its output, as it will classify legal actions as intrusions. The occurrence of this type of error should be minimized so as to grant helpful information to the operators. If too many false positives are produced, the operators will approach to take no notice of the production of the system over time, which may direct to a real intrusion being detected but ignored by the users.

3.2.2 **False negative Error** is the opposite case of false positive. When an actual intrusive action takes place, but system mark it as a non- intrusive activity. False negative errors are harsher than false positive errors because they outputs a deceptive sense of security. When all events are permitted to continue, a suspicious action may be without difficulty passed with connections without intrusion, and will not be brought to the notice of the operator.

3.2.3 **Subversion Error** happens when an impostor modifies the process of the intrusion detector to strength false negatives to occur. rebellion errors are more multifaceted and tie in with false negative errors.

3.3 Problem Scenarios

As discussed in Chapter 1, this thesis research studies that the main focus of my thesis is 'How to select the features (attributes) of the input training data on which learning will be obtained'. Everyone knows that from a set of attributes, not every feature of the training data may be informative or relevant to the detection task and Features may contain false correlations, which delay the process of detecting intrusions. Features may contain false correlations, which delay the process of detecting intrusions. So, a blind choice of feature selection unavoidably include many irrelevant features which may introduce noise and redundancy into the design of classifiers, will definitely do harm to classification performance. It can also increase computation time and can also impact the accuracy of IDS.

I seek an automatic approach so that I can eliminate the manual elements from the development process of IDSs. I take a data-centric point of view and consider intrusion detection as a data analysis process. The central theme of my approach is to apply data mining programs to extensively gathered audit data to compute models that accurately capture the actual behavior (i.e., patterns) of intrusion and normal activities, and get better accuracy. The resultant hierarchical combined detection models are easily adaptable and extensible.

3.3.1 Which Feature Selection method should be used?

For the procedure of attribute creation and feature selection the understanding of the association between features and the classifier accuracy is necessary. Basically, the feature selection procedures can be divided into two types: the filter techniques and the wrapper techniques.

3.3.1.1 Wrapper Model VS Filter Model

The Wrapper approach to feature selection performs a feature space search to evaluate features. Learning algorithm work as part of wrapper model as their evaluation function. The accuracy provided by the Wrappers is usually better accuracy as compare to Filter Model. The reason behind this is only that Filter Model is independent of any learning algorithm, which is advantageous in the sense that it provides better generality and low computational cost. [7], [10]

3.3.2 Which Dataset should be used?

For the purpose to get more realistic results, it was necessary to have real time data to be experimented. To get real dataset, it was not possible for me to observe, analyze and gather a fully real network to get the real values of all attributes. There were many reasons behind it. Such scenario (to get original values from a real network) needs a lot of time. It's also a costly process. It also needs many hardware specifications. And after all this, there still exist chances that my observed data is wrong. The main reason to this is it is not a single person's work. To analyze a whole network and get accurate values needs a full team to work. So, after that I start working on analyzing different datasets used in different papers by different researchers. In [14] 36 datasets of UCI Machine Learning repository are used. Similarly in [3], KDDcup99 dataset is used.

3.4 Focus of Research

3.4.1 Techniques to be focused:

I will use **wrappers model.** The usage of learning model makes this model more efficient. So I can expect high classification performance. The wrapper model consists of two components:

1. Search component
2. Evaluation component.

The main job of search component is to generate parameter settings and feeds its output to the evaluation component.

Since for the arrangement of collection of attribute subsets, mining algorithms are used, the wrapper model have a tendency to provide enhanced performance as feature subsets found are enhanced and appropriate to the prearranged mining algorithm.[8]

To improve accuracy, a **Genetic algorithm** is used as Search Component to produce rules to analyze audit data and extract features that can make a distinction between anomalies

from normal activities. To increases computability I used Rough Tree based architecture as an evaluation component with low computational time and high accuracy.

3.4.2 The KDD Process

The data set used to test the performance of my algorithm will be KDD Cup 99. There are several reasons behind the selection of the data set. First the data set is close to the real scenario which the algorithm is encountered. It is a rich data set as it has a record of millions of records. It has 41 one attributes which are categorical as well as numeric. Due to these salient features of the dataset it is becoming an international standard to evaluate IDS algorithms.

3.4.3 Networking Attacks

Here, I will explain four major categories of networking attacks analyzed by KDD Cup 99. Every attack on a network can comfortably be sited into one of these groupings [3].

3.4.3.1 Denial of Service (DoS):

A DoS attacks is a type of hits in which the attacker creates a software, hardware e.g. memory too tiring or too full to provide lawful networking requests and hence stops users to get the privileges to a particular machine.

3.4.3.2 Remote to User attacks (R2L):

A remote to user attack is an attack in which a user transfers packets to a machine over the internet, which she or he does not have approachability to exploit privileges which a local user would have on the computer.

3.4.3.3 User to Root Attacks (U2R):

These attacks are the types of violations in which the hacker begins off on the system with a standard user account and tries to misuse vulnerabilities in the system in order to gain maximum user privileges.

3.4.3.4 **Probing:** Probing is a kind of violence in which the hacker scrutinizes a machine or a networking mechanism in order to decide weaknesses or vulnerabilities that may later be broken so as to compromise the system. If a connection doest not lie in the above categories, then it means it is a normal connection.

Chapter 4
Design

4. System Design

The purpose of design is to create architecture for the developing implementations. Object oriented design is a technique of design about the procedure of objects oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of system under design.

The design phase focuses on defining the software to implement the application. The design object is to create a model of the system, which can be used later to build the system .The design goal is to find the most excellent possible design within the limitations imposed by the requirement and the physical and social environment in which the system will manage.

4.1 Introduction

The goal of the project is to implement an efficient algorithm using different types of techniques to predict and classify the features.

The first and foremost task was what type of technique would give us maximum information therefore implementation of proposed algorithm needs a deep study of the techniques chosen to get better accuracy and improvement.

4.2 Design Requirements

4.2.1 Genetic Algorithm

Genetic Algorithms (GAs) have been effectively applied to respond search and optimization problems. The vital idea of a GA is to search an postulation space to find the optimum assumption. The first step is generating a pool A pool at random and finally compute a fitness function. Thus depending upon the values of fitness function, the genetic algorithm is terminated. If better value of finess is computed, that value is taken as a standard for GA algorithm. Different fitness functions are evaluated and hypothesis is tested for each generation.

- **Genetic Algorithm:**

The requirement of the algorithm is to generate a set of rules for misused detection. Here, one rule is represented by a single chromosome.

- Start

Initial population on *n* chromosomes, is randomly generated. Each Chromosome has 41 genes i.e. 10101011110100100011010110110100101010 1101

- Fitness

Step-I: Make the discernibility matrix and searching cores by using the following formula: Finding the reducts of attributes for feature reduction is really a very complex task and can be said NP-hard but a discernibility matrix is used to reduce the computation intensity. An nxn matrix with entries Cij define as

$$(c_{ij}) = \{a \in Q : a(x_i) = a(x_j)\}$$

$$D(x_i) = D(x_j) \quad i, j = 1, 2, \ldots, n$$

where a is condition attribute and D is a decision attribute.

To explain whole process lets i=5, j=3...now check D(5) = 2 and D (3)=1 , both are not equal ,so D will be member of Cij..

now check a(5) = 2, a(3) = 1, which are again not equal..so **a** will be entered in Cij..

now check b(5) = 3, b(3) = 2, which are again not equal..so **b** will be entered in Cij..

now check c(5) = 2, c(3) = 2, which are equal..so **c** will be not be entered in Cij..

now check d(5) = 2, d (3) = 3, which are again not equal..so **d** will be entered in Cij..

so, In C_{15} = {a,b,d,D} Using same process finalize all the entries of discernibility matrix.

Discernibility matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2	c,d											
3	d	c,d										
4	a	a,c,d	a,d									
5	a,b,d,D	a,b,c,D	a,b,d,D	b,d,D								
6	b	b,c,d	b,d	a,b	a,d,D							
7	c,D	d,D	c,d,D	a,c,D	a,b,c,d	b,c,D						
8	a,b,c,d	a,b,c	a,b,c,d	b,c,d	c,D	a,c,D	a,b,c,d,D					
9	d	c	d	a,d	a,b,D	b,d	c,d,D	a,b,c				
10	b,c,d	b	b,c,d	a,b,c,d	a,b,c,D	b,c,d	b,d,D	a,b,c	b,c			
11	a,b,d,D	a,b,c,D	a,b,d,D	b,d,D	b	a,b,d,D	a,b,c,d	b,c,D	a,b,D	a,c,d		
12	b,d	b,c,d	b	a,b,d	a,b,d,D	b,d	b,c,d,D	a,b,c,d	b,d	c,d	a,d,D	

. b,c,d	3	1/3	1/3
. b,d	2	0	½
. c,d	2	½	½
. a,c,d	3	1/3	1/3
. b,c,d	3	1/3	1/3
. a,b,c	3	1/3	0
. c	1	1/1	0
. b	1	0	0
. b,c,d	3	1/3	1/3
. a,d	2	0	½
. b,d	2	0	½
. a,b,c,d	4	¼	¼
. d	1	0	1/1
. b,c,d	3	1/3	1/3
. b	1	0	0
. a,b	2	0	0
. b,c,d	3	1/3	1/3
. a,d	2	0	½
. a,b,c,d	4	¼	¼
. a,b,d	3	0	1/3
. a,b,c,d	4	¼	¼
. b	1	0	0
. b,d	2	0	½
. b,c,d	3	1/3	1/3
. b,d	2	0	½
. a,b,c,d	4	¼	¼
. a,b,c	3	1/3	0
. a,b,c	3	1/3	0
. b,d	2	0	½
. c,d	2	½	½
. a,c,d	3	1/3	1/3
. b,c	2	½	0
. a,b,c,d	4	¼	¼

$$CC_n(C) = -(\frac{1}{2}+\frac{1}{4}+\frac{1}{3}+\frac{1}{2}+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}+\frac{1}{1}+\frac{1}{3}+\frac{1}{4}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}+\frac{1}{4}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}+\frac{1}{2}+\frac{1}{2})$$
$$= -8.17$$

CCn (d) = -(½ + + 1/1 + ¼ + 1/1 +1/3 + ½ +½ + 1/3 + 1/3 +1/3 +½ +1/2 + 1/4 + 1/1 + 1/3 +1/3 +½ +¼ +1/3 +¼ +½ +1/3 +½ +¼ +½ +½ +1/3 +¼) = -12.5

Now compute the value of CC_p (c)

- Select all those entries of Cij whose value for D (i) and D(j) are not same i.e. C_{54} = {b,d,D}, $C_{10,5}$= {a,b,c,D} etc

From the above selected set, select only those entries which contain c as their part. i.e. from above example C54 = {b,d,D}, C10,5= {a,b,c,D} , C54 will not be selected.

Now calculate their probability i.e. probability of C10,5= {a,b,c,D} will be 1/3. because probability must be only for conditional attributes.

Similarly calculate the probabilities of all rest values

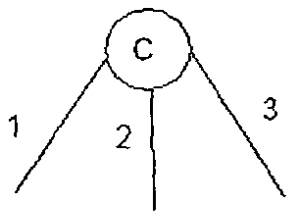
Attributes with D(Positive Cases)	**Attribute length (With out D)**	**Probability of 'c'**	**Probability of 'd'**
. a,b,d,D	3	0	1/3
. c,D	1	1/1	0
. a,b,d,D	3	0	1/3
. a,b,c,D	3	1/3	0
. d,D	1	0	1/1
. a,b,c,D	3	1/3	0
. a,b,d,D	3	0	1/3
. c,d,D	2	½	½
. a,b,d,D	3	0	1/3
. c,d,D	2	½	½
. b,d,D	2	0	½
. a,b,D	2	0	0
. a,c,D	2	½	0
. b,d,D	2	0	½
. a,c,D	2	½	0
. b,d,D	2	0	½
. a,d,D	2	0	½
. c,D	1	1/1	0
. a,b,D	2	0	0
. a,b,c,D	3	1/3	0
. a,b,d,D	3	0	1/3
. b,c,D	2	½	0
. a,b,d,D	3	0	1/3
. a,b,c,d,D	4	¼	¼
. b,c,d,D	3	1/3	1/3
. b,c,D	2	½	0
. a,d,D	2	0	1/2

CCp (c) = 1/1+1/3+1/3+1/2+1/2+1/2+1/2+1/1+1/3+1/2+1/4+1/3+1/2 = 7.58

CCp (d) = 1/3+1/3+1/1+1/3+1/2+1/3+1/2+1/2+1/2+1/2+1/2+1/3+1/3+1/4+1/3+1/2

= 7.08

$CC_T(c) = 7.58 - 8.17 = -0.58$



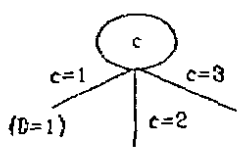
1. First calculate the possible outcomes of 'c' from input matrix, which are 1,2,3..
2. now ignore attributes a,b,d and save rest of the table from input matrix in this way

	. c	D
0	2	1
1	3	1
2	2	1
3	2	1
4	2	2
5	2	1
6	3	2
7	1	1
8	2	1
9	3	1
10	2	2
11	2	1

Now observe the values of c

- When c=1

First see, there is only one cell containing 1 as input on 7th index. And also D =1 in 7th index. So it is clear whenever c=1, also D=1.



No.	c	D
1	2	1
2	3	1
3	2	1
4	2	1
5	2	2
6	2	1
7	3	2
8	1	1
9	2	1
10	3	1
11	2	2
12	2	1

- When c=3

Now check other attributes for c=3.

Save the values of all attributes when c=3.

3	2	2	3	2	2
4	2	1	3	1	1
5	2	1	2	2	1
6	2	2	1	2	2
7	2	1	1	3	1

Make discernibility matrix

	0	1	2	3	4	5	6	7
0	-							
1	. d	-						
2	. a	. a,d	-					
3	. a,b,d,D	. a,b,d,D	. b,d,D	-				
4	. b	. b,d	. a,b	. a, d , D	-			
5	. d	. d	. a,d	. a,b, D	. b,d	-		
6	. a, b,d,D	. a,b,d,D	. b,d,D	. b	. a,b,d,D	. a,b,D	-	
7	. b,d	. b	. a,b,d	. a,b,d,D	. b,d	. a,d, D	.a,d,D	-

As there is no such pairs of D from which Core Attribute can be selected as Core.

If there is no Core Attribute or More then one Attribute then measure CCT (ak) for each attribute in reduct set and select condition attribute with maximum value to be a node.

CCp (ak) for attributes with D

	Length	{a}	{b}	{d}
{a,b,d,D}	3	1/3	1/3	1/3
{a,b,d,D}	3	1/3	1/3	1/3
{a,b,d,D}	3	1/3	1/3	1/3
{a,b,d,D}	3	1/3	1/3	1/3
{b,d,D}	2	0	½	½
{b,d,D}	2	0	½	½
{a,d,D}	2	½	0	½
{a,b,D}	2	½	½	0
{a,b,d,D}	3	1/3	1/3	1/3
{a,b,d,D}	3	1/3	1/3	1/3
{a,b,D}	2	½	½	0
{a,b,D}	2	½	0	½
	Sum	4	4	4

CCn (ak) for attributes without D:

	Length	. a	. b	. d
{d}	1	0	0	1
{a}	1	1	0	0
{b}	1	0	1	0
{d}	1	0	0	1
{b,d}	2/	0	½	½
{a,d}	2	½	0	½

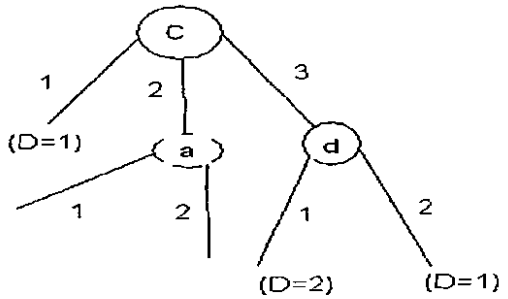
{b,d}	2	0	½	½
{d}	1	0	0	1
{b}	1	0	1	0
{a,b}	2	½	½	0
{a,d}	2	½	0	½
{a,b,d}	3	1/3	1/3	1/3
{b}	1	0	1	0
{b,d}	2	0	½	½
{b,d}	2	0	½	½
{b,d}	2	0	1/2	½
	Sum	2.83	4.33	6.83

$CC_T(a) = CCp(a) - CCn(a) =$ 4 - 2.83 = 1.17

$CC_T(b) = CCp(b) - CCn(b) =$ 4 - 4.33 = -0.33

$CC_T(c) = CCp(c) - CCn(c) =$ 4 – 6.83 = -2.83

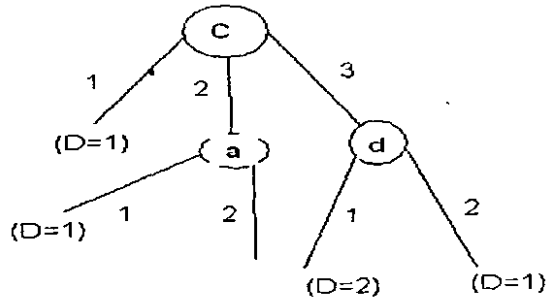
As 'a' has greatest CC_T, So **'a' will be selected as Core Attribute.**



Here the possible outcomes of 'a' are 1, 2.Now select a, D from the input matrix

. a	D
1	1
1	1
2	1
2	2
1	1
1	1
2	2
1	1

Here, whenever a=1, D=1, and a=2, D=1, 2



Now for a=2

	. a	. b	. d	D
0	2	2	1	1
1	2	3	2	2
2	2	1	2	2

Discernibility Matrix of above matrix

	0	1	2
0			
1	{b,d,D}		
2	{b,d,D}	{b}	

Here again more then one core attribute exists. So calculate CC_T and CC_N

CC_T:

	Length	. b	. d
{b,d,D}	2	1/2	½
{b,d,D}	2	½	½
Sum		1	1

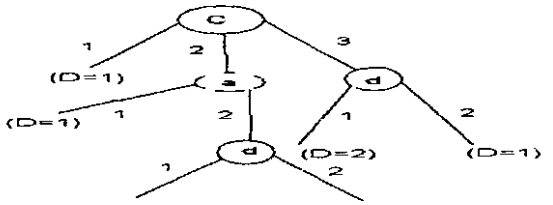
CC_N:

	Length	. b	. d
{b}	1	1	0
Sum		1	0

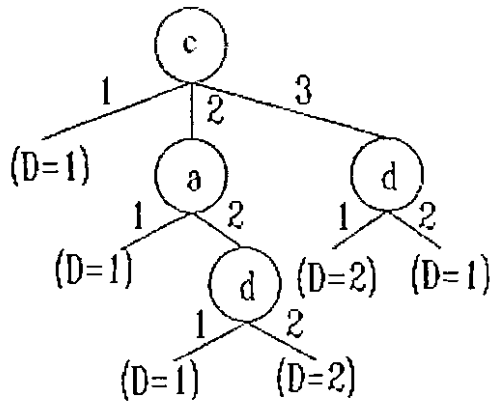
$CC_T(b) = CCp(b) - CCn(b) = 1 - 1 = 0$

$CC_T(d) = CCp(d) - CCn(d) = 1 - 0 = 1$

As 'd' has greatest CC_T, So 'd' will be selected as Core Attribute.



. d	D
1	1
2	2
2	2



4.2.3 KDDcup99 Dataset

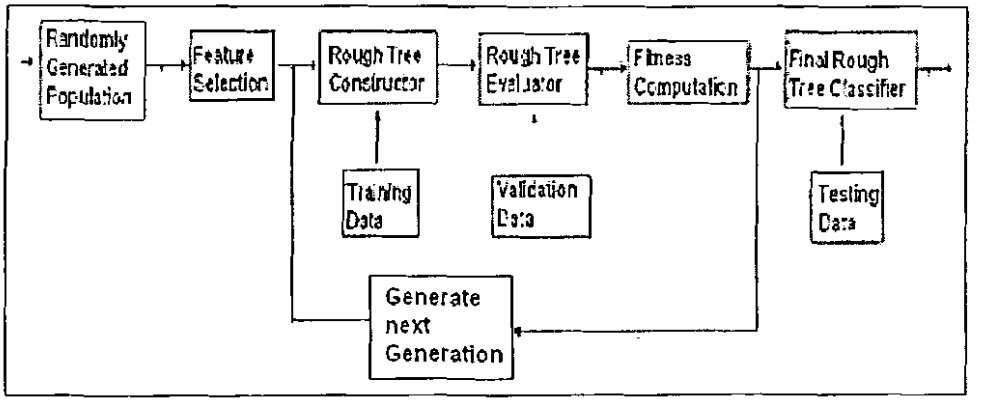
4.2.3.1 KDD Cup 99 Information

The main objective of the classifier knowledge contest was to discover an analytical model (i.e. a classifier) able of feature among genuine and illegal connections in a computer network structured in concurrence with the KDD'99 conference.

4.2.3.2 Intruder Detection Learning

Algorithm to discover network intrusions shelters a computer network from not permitted users, counting possibly insiders. The intrusion detector-learning job is to construct a analytical model or algorithm/structure capable of distinctive between ``bad" quires which are considered as the intrusions or the external attacks and rest of the packets are called the normal quires which does not produce any harm to the network.

4.3 Reference Architecture



The overall system architecture is designed to outperform the rough tree algorithm without feature selection. This hybrid technique is able to focus on relevant features and eliminate unnecessary, non-relevant or noisy features. In my algorithm, Genetic Algorithm is the search component and Rough Tree is the evaluation component. A detailed description of this algorithm is shown in above Figure. The initial population is randomly generated dataset 'KDD cup 99'. Every chromosome has 41 genes, each of which represents a feature of the input data. If it is 1 then it means the represented feature is used during constructing rough trees; if it is 0 then it shows the feature is not providing its contribution. For each individual in the current population, a rough tree is built using the Rough Tree Algorithm. Then this resulting rough tree is tested over nine validation data sets, which generates nine classification error rates. To calculate the fitness of this

individual, add up all of these classification error rates. The lower the classification error rate, the better the fitness of the individual.

Once e you have complete computing the fitness values of all individuals of the current population, the GA begins to generate next generation as follows:

(1) Use graded for each input attribute.

(2) Swap genes between parents.

(3) Create a bit level mutation.

I set the maximum number of generations equal to 100.The procedure above is iteratively executed until the last limit of generation. Finally, the best individual of the last generation is selected and a final Rough Tree is constructed through rough tree classifier. After that finally the test data is applied on it for testing.

4.4 Algorithm and Functions

4.4.1 Function GA ( )

The algorithm I have used for intrusion detection is Genetic algorithm.
The algorithm has following steps:

Take R input for number of chromosomes.
Generate randomly R number of chromosomes
    Divide full training data into 10 files
    Select first 10 % data as training data
    Place rest of 9 chunks of data as 9 validation sets
For each K number of iterations
    Do
    Read each line from training set
    Select the features
    Draw a rough tree for each individual of the training data.
    Evaluate the rough tree by applying validation data one by one
    Calculate the minimum error rates of each validation data

    Sort errors in ascending order
    End
    If minimum error found
        Then
            Add all minimum error rates
            And consider it as fitness value
        Else recalculate;
    If fitness value is lesser then upcoming error rates
Then draw final rough tree classifier
Else

    Discard upper half of errors
    Randomly pick the two rules and cross them to each other and these yields two extra chromosomes (rules).

Include them to the regulations.
Generate next generation

4.4.2 Function RT ( )

RoughTree Algorithm

RoughTree(Att, Data, threshold)
INPUT: Att- a set of attributes
Data- training data
threshold- a threshold for dependence
$P_i=\{a_i, d\}$ (d is the decision attribute, $a_i \in$ Att)
Q_i=Att-P_i
OUTPUT: a RoughTree
BEGIN
Step1: Create a tree node, load the Data, compute all $K(P_i, Q_i)$ according to (6);
Step2: Check the stopping criteria:
(1) instances number <30;
(2) instances number / number of branches <50;
(3) $Max(K(P_i, Q_i))$<threshold;
If one of the criteria is satisfied, go to step4;
Else, go to step3;
Step3: Index=arg max$K(P_i, Q_i)$, choose the attribute a_{Index} to split the data, Att=Att-a_{Index}, go to step1;
Step4: In the current node, mark the node as a leaf;
build a Naive Bayesian classifier from Data in each leaf.
END.

Chapter 5
Implementation

5. Implementation

5.1 Dataset

The data set used was 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. The competition objective was to construct a network intrusion detector, a model which can predict and capable of seperating the "bad" connections, called intrusions or attacks, from "good" normal connections. This database holds a normal set of data to be reviewed, which includes a wide variety of interruption replicated in a military network environment.

5.1.1 Types of Attack

1	Probe -
2	Denial of Service (DOS)
3	User-to-Root (U2R)
4	Remote-to-Local (R2L)

5.1.1.1 Classification of Intrusions:

These are the four main categories which are further subdivided into sub categories

Probe	DOS	U2R	R2L
Ipsweep	Back	Buffer_overflow	Ftp_write
Nmap	Land	Loadmodule	Guess_passwd
Portsweep	Neptune	Perl	Imap
Satan	Pod	Rootkit	Multihop
	Smurf		Phf
	Teardrop		Spy
			Warezclient
			Warezmaster

5.1.1.2 Derived Features from Dataset

There are different types of features depending upon the errors of attacks. These are summarized in the tables below:

Table 1: Features of individual TCP connections [12]

Feature Name	Description	Type
Duration	Length(seconds) of the connection	Continuous
Protocol Type	Type of protocol (Tcp, UDP	Discrete

	etc)	
Service	Destination service of Network from source	Discrete
Src_bytes	Number of bytes sent from source to destination	Continuous
Dst_Bytes	Numebr of byetes sent from destination to source	Contnuous
Flag	Status (normal or error)	Discrete
Land	=1 if connection is from same host port else =0	Discrete
Wrong_fragment	No. of wrong packets sent or received	Continuous
Urgent	Number of urgent data fragments	Contiouous

Table 2: Content features within a connection [12]

Feature Name	Description	Type
Hot	Number of hot indications	Continuous
Num_Failed_Logins	Number of failed login attempts	Continuous
Logged_in	=1 if successfully logged in else =0	Discrete
Num_compromised	No. of those conditions which are compromised	Continuous
Root_shell	=1 if it is a root else=0	Discrete
Su_attempted	=1 if su root command Else =0	Discrete
Num_root	Number of kernel priviliges	Continuous
Num_File_Creat	Files generations	Continuous
Num_Shells	Num of shells (local)	Continuous
Num_Access_Files	Number of opterations accesses	Continuous
Num_outbound	This is for File Transfer Control Protocol outbounds	Continuous
Is_guest	=1 if login is a guest	Discrete

Table 3: Traffic features [12]

Feature Name	Description	Type
Count	Current connections same hosts	Continuous
Serror_Rate	No. of SYN faults	Continuous
Rerror_Rate	NO. of REJ faults	Continuous
Same_SRV	No. of same connections	Continuous
Diff_SRV	No. of different connections	Continuous
SRV_Count	No. of Connections with same service	Continuous
SRV_serror	No. of SRV errors	Continuous
SRV_rerror	No. of REJ errors	Continuous
SRV_diffi_host	No. of diffent hosts	Continuous

5.1.2 Datasets Management

How to manage such a rich Dataset

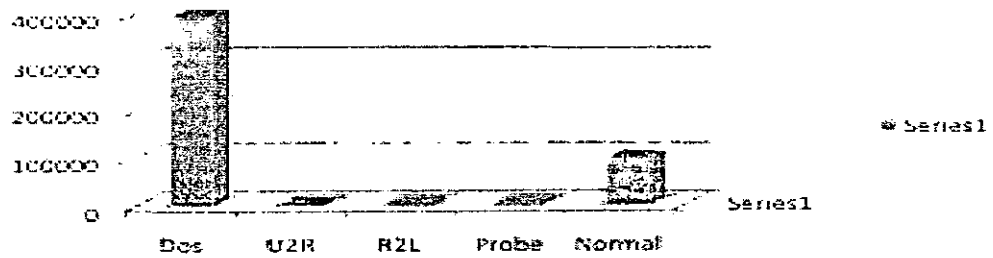
Step I: I used the 10 % of the KDDCUP/99 training data and full testing data (311029/99 cases) to trial.

1. Divide the training data and testing data into many minor training datasets and testing datasets according to 4 intrusion categories (Probe/DOS/U2R/R2L).
2. Now for each intrusion group, divide its training data into 10 secede files of equal volume.
3. Select one file as it s training dataset and rest 9 as validation datasets.

   The training dataset and test dataset for DOS includes all DOS attacks and normal cases in the original training data and test data. Means except DOS attacks all the rest attacks will be considered as normal records.

5.1.2.1 Some statistics about dataset

After separating the training data in 5 catagories, DOS contains 391460 rows, U2R contains 60 rows, R2L contains 1130, Probe contains 4110 and Normal contains 97280 rows. This data distribution is explained in the following bar satatistics figure. But after deviding each file into 10 files, each file contains different divisions of data



5. Environments

5.2.1 Java Eclipse

The JDT project provides the tool plug-ins. These plug-ins implements a Java Integrated Development Environment, which supports the development of any Java application, it also provides supportive Eclipse plug-ins. JDT adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, editors, wizards,

builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself. In my project Java Eclipse is used to implement whole algorithm.

5.2.2 Matlab 7

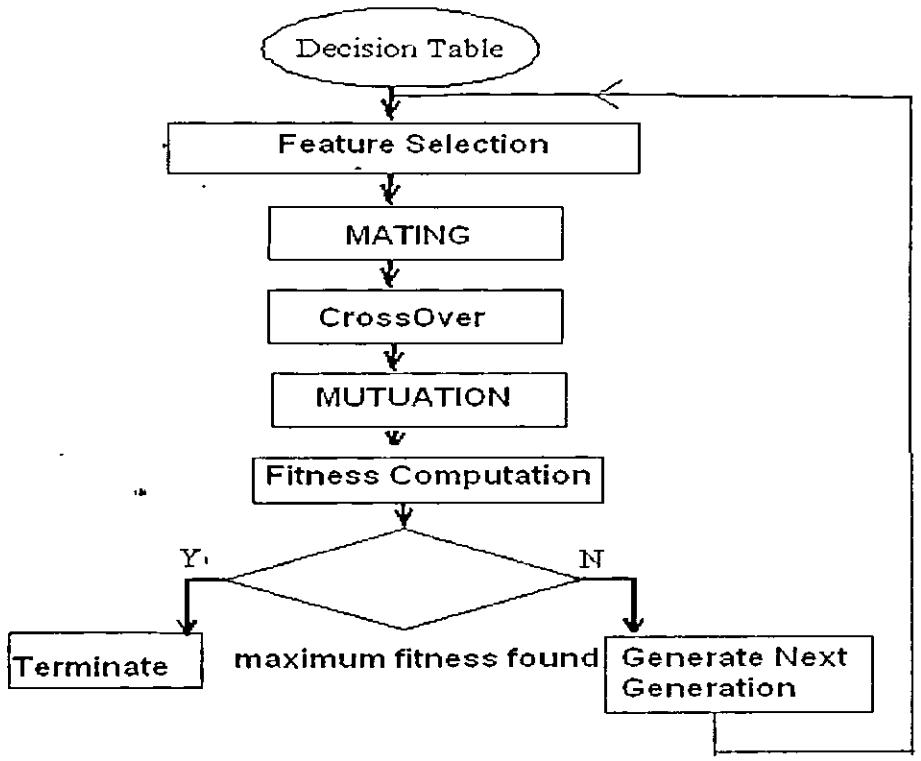
MATLAB® is a high-performance language for technical computations. It provides integrity among computation, visualization, and programming in an easy-to-use environment where troubles and solutions are articulated in well-known mathematical notation. I have splitted my full dataset my programming in Matlab 7.0.

5.2.3 Graphviz 2.24

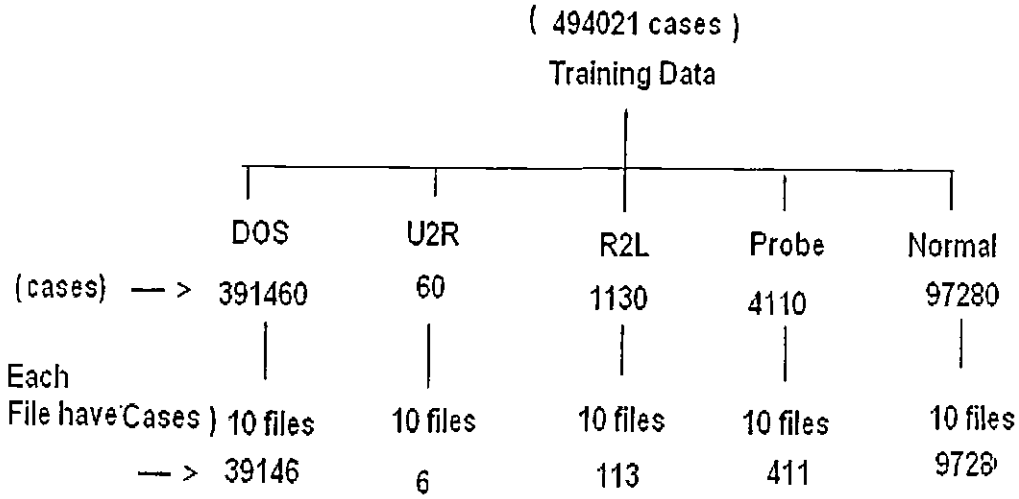
GVedit is a Graphviz tool for creating, viewing, editing and processing DOT files. It allows users to set attributes of graphs with dialogue boxes and save them for future use. GVedit users can also take benefit of the easy instant previewing feature of the program.

5.3 Flow Controls:

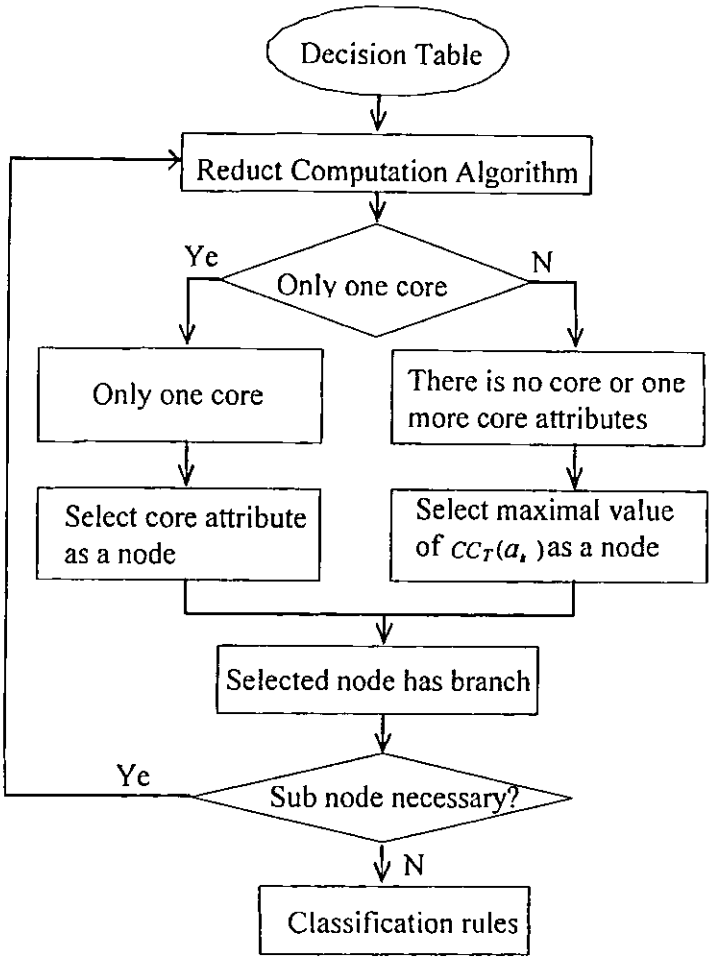
5.3.1 Genetic Algorithm



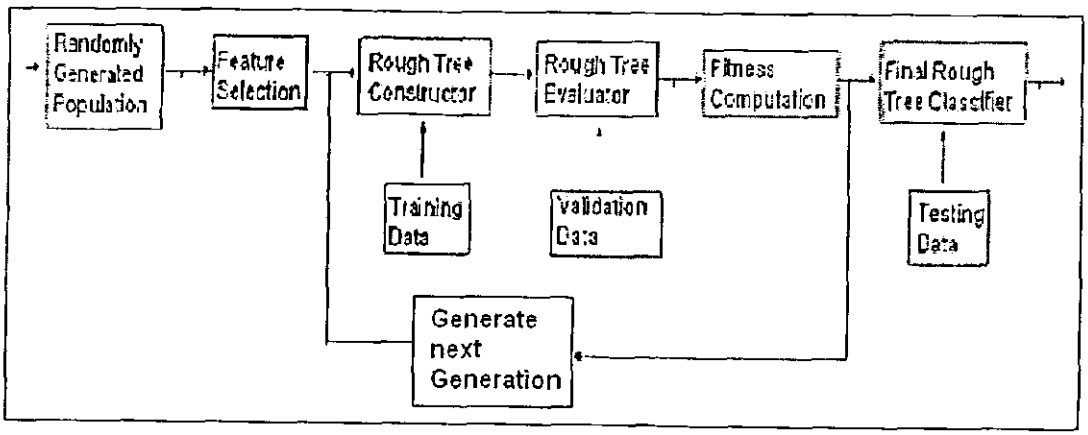
5.3.2 Dataset



5.3.3 Rough Tree Algorithm



5.4 Summary:



5.5. Implementation Code

ROUGH TREE

```
                    /*********************************/
                    /*** STEP I: Read Input Data ***/
                    /*********************************/
String inputFileName=
      "F:/eclipse Work/Roughset_Tree/roughtree_data.txt";
String outputFileName=
 "F:/eclipse Work/Roughset_Tree/discernibility matrix_data.txt";
System.out.println("ROUGHT SET \n\n ---Step I--- \n\t\t ---INPUT MATRIX-
--");       // Create FileReader Object
FileReader inputFileReader = new FileReader(inputFileName);
FileWriter outputFileReader = new FileWriter(outputFileName);
            // Create Buffered/PrintWriter Objects
BufferedReader inputStream   = new BufferedReader(inputFileReader);
PrintWriter    outputStream  = new PrintWriter(outputFileReader);

                    /*********************************/
          /******* ASCII code conversion(char to int) ********/
                    /*********************************/

if(array[rw][cl]==48)
      {
      value=0;
      }
else if(array[rw][cl]==49)
      {
      value=1;
      }
else if(array[rw][cl]==50)
      {
      value=2;
      }
else if(array[rw][cl]==51)
      {
      value=3;
      }
else if(array[rw][cl]==52)
      {
      value=4;
```

```
        }
else if(array[rw][cl]==53)
        {
        value=5;
        }
else if(array[rw][cl]==54)
        {
        value=6;
        }
else if(array[rw][cl]==55)
        {
        value=7;
        }
else if(array[rw][cl]==56)
        {
        value=8;
        }
else if(array[rw][cl]==57)
        {
        value=9;
        }
        matrix[rw][cl]=value;

                /*********************************/
              /*** STEP II -- DISCERNIBILITY MATRIX ***/
                /*********************************/

public static  char[][][] disc(int [][] input_matrix, int cols, int
rows)
{
char disc_matrix[][][];
disc_matrix= new char[cols][rows][rows];

char attr[]= {'a','b','c','d','D'};

char selected_attr[];
selected_attr= new char[1000];

int last=cols-1;
int pointer=-1;
int rrr=0;
int ccc=0;
int t=0;
int d=0;

for(pointer=0;pointer<rows;pointer++)
{
   last=cols-1;
   rrr=0;
   ccc=0;
   t=0;
   d=0;
      for (rrr=0;rrr< rows; rrr++)
      {
      for(ccc=0;ccc<cols;ccc++)
            if (input_matrix[pointer][ccc]==input_matrix[rrr][ccc])
                  {
                  }
            else
                  {
                  selected_attr[t]=attr[ccc];
```

```
                    if(d<cols)
                    {
                    disc_matrix[d][rrr][pointer]=attr[ccc];
                    d++;
                    }
                    t++;
                    }
              if (rrr<rows)
              {
              d=0;
              }
          }
        if(pointer==rows-1)
          {
            for(int rw=0;rw<rows;rw++)
            {
                for(int cl=0;cl<rows;cl++)
                    {
                       for(int dd=0;dd< cols;dd++)
                          {
                             if((rw==cl)||(rw>cl))
                             {
                             disc_matrix[dd][cl][rw]= 00; // null=00
                             }
                          }
                    }
              }
          }
        }
return disc_matrix;
}
            /*************************************/
/*** Step III: Select Negative cases from Discernibility Matrix ***/
            /*************************************/

public static  char[][] conditional_rows (char [][][] disc_matrix, int
cols, int rows)
{
      char condition_attr[][];
      condition_attr= new char[1000][6];

      char [][] conditionattribute;
      conditionattribute = new char[1000][6];
      int frows=0;
      int fcols=0;
      char nullntfound[];
      nullntfound=new char[1000];
      int counter=0;
      char temp=00;
      int p=0;
      int d=0;
      int ii=0;
      int jj=0;
for(int rww=0;rww<rows;rww++)
    {
      first:
            for(int clw=0;clw<rows;clw++)
                {
                 fcols=0;
                 {
                 for(d=0;d<cols;d++)
```

```
                    {
                    if (((disc_matrix[d][clw][rww])=='D')&&(clw<rows))
                          {
                          continue first;
                          }
                    if(clw==rows)
                          break;
                    }
                   }
      if(clw!=rows)
        {
           for(int dd=0;dd< cols;dd++)
             {
                    if(disc_matrix[dd][clw][rww]!=00)
                          {
                          temp=disc_matrix[dd][clw][rww];
                          conditionattribute[frows][fcols]=temp;
                    if(dd==0)
                          if(conditionattribute[frows][dd]!=00)
                    counter=counter+1;
                    fcols++;
                    }
                    if(dd==cols-1)
                    frows++;
                    }
                    }
            }
        }
for(int i=0;i<100;i++)
    {
      for(int j=0;j<cols;j++)
          {
            if(j==0)
                  if(conditionattribute[i][j]==00)
//if null value exist at first index then goto next row
                  {
                  break;
                  }
                  if( conditionattribute[i][j]!=00)
                        {
                        while(jj<cols)
                              {
                        condition_attr[ii][jj]=conditionattribute[i][jj];
                              p++;
                              jj++;
                              }
                        if (jj==5)
                              {
                                    ii++;
                                    jj=0;
                              }
                        break;
                        }
                  }
      return condition_attr;
    }
            /*************************************/
/*** Step IV: Select Positive cases from Discernibility Matrix ***/
            /*************************************/
```

```
public static  char[][] decisional_rows (char [][][] disc_matrix, int
cols, int rows)
{
      char temp_att[];
      temp_att= new char[1000];

      int t=0;
      int tt=0;
      int d=0;
      int c=0;

      char decision_attr[][];
      decision_attr=new char[1000][6];
for(int rww=0;rww<rows;rww++)
 {
  for(int clw=0;clw<rows;clw++)
    {
      for(int dd=0;dd< cols;dd++)
         {
           if(disc_matrix[dd][rww][clw] == 'D')
              {
            tt=tt+1; //returns number of rows containing D as their part
            row_length[c]=dd+1;// calculate the length of each row
            while(d<=dd)
                  {
                        temp_att[t]=disc_matrix[d][rww][clw];
                        decision_attr[t][d]= temp_att[t];
                        d++;
                  }
                  d=0;
                  t++;
                  c++;
              }
         }
      }
}
return decision_attr;
}
            /*************************************/
            /*** Step V: Select CORE Attribute ***/
            /*************************************/
public static  char[] core(char [][][] disc_matrix, int cols, int rows)
      {
        char temp_att[];
        temp_att= new char[1000];

        char tmp[];
        tmp=new char[1000];

        char core_att[];
        core_att= new char[1000];

        int row_length[];
        row_length= new int[1000];

        int row_index[];
        row_index= new int[1000];

        int t=0;
        int tt=0;
        int d=0;
```

```
        int c=0;

        char decision_attr[][];
        decision_attr=new char[1000][6];
             for(int rww=0;rww<rows;rww++)
                {
                     for(int clw=0;clw<rows;clw++)
                        {
                           for(int dd=0;dd< cols;dd++)
                              {
                                 if(disc_matrix[dd][rww][clw] == 'D')
                                       {
                                        tt=tt+1;
//returns total number of rows containing D as their part
                                        row_length[c]=dd+1;
// calculate the length of each row
                                      while(d<=dd)
                                        {
                                   temp_att[t]=disc_matrix[d][rww][clw];
                                   decision_attr[t][d]= temp_att[t];
                                   d++;
                                        }
                                   d=0;t++;c++;
                                        }
                                   }
                           }
                  }
                  /*******************************/
             /*** calculate the minimum row length ***/
                  /*******************************/
       for (int y=0;y<tt-1;y++)
             {
             if(row_length[y+1]<row_length[y])
                  {
                        templ=row_length[y+1];
                        if(templ<temp)
                        temp=templ;
                  }
             }
 for (int y=0;y<tt;y++)
             if(row_length[y]==temp)
             {
             k++;
             }
             k=0;
             System.out.println("\n\t Core Attributes are:");
                for(c=0;c<tt;c++)
                        {
                           for(d=0;d< cols;d++)
                              {
                                    if(row_index[k]!=0)
                                    {
                                          row= row_index[k];
                                          tmp[k]=decision_attr[row][d];
                                          break;
                                    }
                              }
                              k++;
                        }
for(c=0;c<cols;c++)
      {
```

```
            if(tmp[c]!=00) //check cells which contain null value
            {
                  core_att[cc]=tmp[c];
                        if(core_att[cc]!=00)
                              cc=cc+1;
             }
      }
            return core_att;
      }
            /*************************************/
     /*** Step VI: Calculate Classification Contribution ***/
            /*************************************/

public static  double[] classification__contribution_N(char []
core_attr,char [][] con_rows,int cols,int Crows, double[] Clength)
      {
        double nn[];    // length of each conditional row
        nn= new double[Crows];

        double CCp=0;
        char core=00;
        int i=0;
        int j=0;
        int k=0;
        int len=0;
        double temp=0;
        double inv=-1;
while(i<cols)
      {
      if (core_attr[i]==00)
      {
      len=i;
      break;
      }
      i++;
      }
for(i=0;i<len;i++)
      {
      core=core_attr[i]; //select core attribute for CC
      CCp=0.0;
      for (j=0;j<=Crows;j++)
            {
            for(k=0;k<cols;k++)
                  {
                  if(con_rows[j][k]==core)
                        {
                        temp=Math.pow(Clength[j],inv);
                        CCp=CCp+ temp;
                        nn[i]=CCp;
                        }
                  }     //end of loop with k
            }     //end of loop with j
System.out.println(" CCn ( "+ core + " ) = " + nn[i]);
      }     //end of loop with i
      return nn;

            /*************************************/
  /*** Step VII: Calculate Total Classification Contribution ***/
            /*************************************/
```

```
public static  double[] classification__contribution_T(char []
core_attr,double [] CCp, double [] CCn,int ln )
{
      double tt[];      // length of each conditional row
      tt= new double[ln];
      for(int i=0;i<ln;i++)
        {
      tt[i]=Math.round(CCp[i]-CCn[i]);
      System.out.println(" CCt ( "+ core_attr[i] + " ) = " + tt[i]);
        }
        return tt;
      }

          /*************************************/
        /*** Step VIII: Select Root of Rough Tree ***/
          /*************************************/

public static  char Root_Node(char [] core_attr, double [] CCt,int ln )
      {
            int max=0;
            char root;
            for(int i=0;i<ln;i++)
            {
                  if((i+1)<ln)
                  {
                        if(CCt[i]>CCt[i+1])
                              {
                                    max=i;
                              }
                  }
            }
            root= core_attr[max];
            System.out.println("\n\n\t Root Node: "+ root);
            return root;
      }
```

Chapter 6
Result

6. Results

This research focuses on the efficient utilization of making rough theory based decision tree for the classification of computer networks attacks. Here 10 percent of the KDDCUP99 data is used for training and rest of the data is used for testing the GA based rough tree for intrusion detection systems. Different parameters have been utilized for analyzing my model. For each intrusion a separate tree is designed using rough set theory. In order to check the applicability of our model the KDDCUP99 data set has been split into four small training datasets and four testing data sets i.e. Probe/Dos/R2L/U2R. For each intrusion training data set, each data set is split into 10 small training datasets. One is used as a training and rest are used as a validation set. A GA based rough tree is build for each type of intrusion and different experiments have been performed. The genetic algorithm is randomly initialized and for each generation, each gene shows whether a specific feature should be utilized or not. One the basis of the selected features, the fitness value is calculated and Rough tree is finally selected.

6.1 Cross Validation and Testing Error Curves for Different Intrusion Attacks

The validation and testing error curves for different intrusion attacks are shown in Figure 6.1-Figure 6.4. The errors for Probe attacks were converged in 61 generations. The maximum accuracy for Dos was achieved at 45 generations. While for R2L and U2R the error graph was converged at 111 and 88 generations respectively. Although the cross validation error for Probe at 35^{th} generation was less than 5% but the testing error has an increasing trending which shows that rough tree at 35^{th} generation is less reliable than that of 66^{th} generation. In Figure 6.1 – Figure 6.4 the different generations are sorted in decreasing order and the last generation at each graph shows the maximum achievable error results (in percentage). The DOS and U2R intrusions classification achieved minimum classification / prediction error.

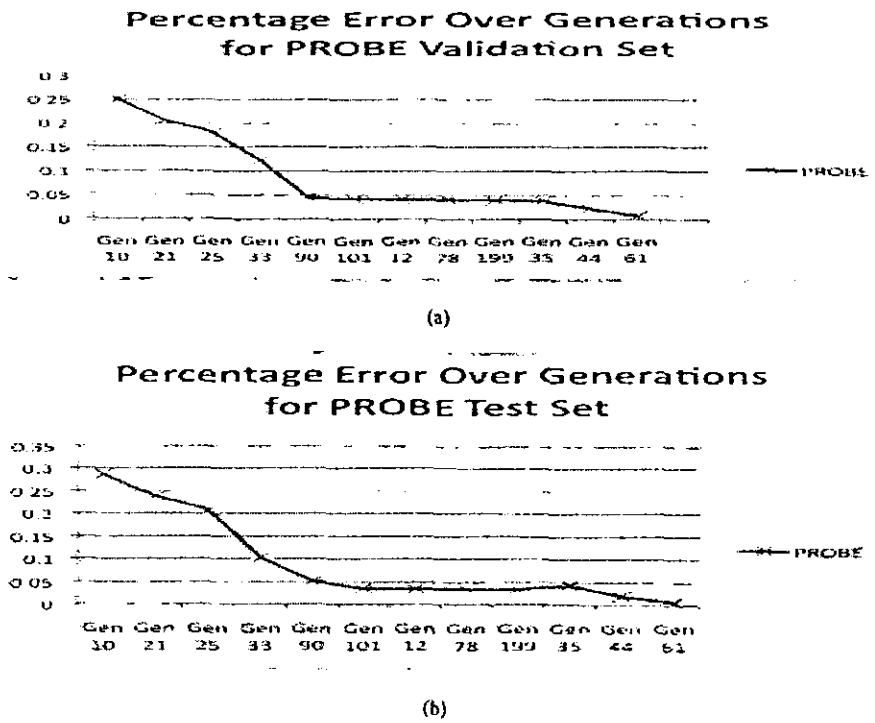


Fig. 6.1. Percentage error (a) validation data (b) testing data for PROBE detection.

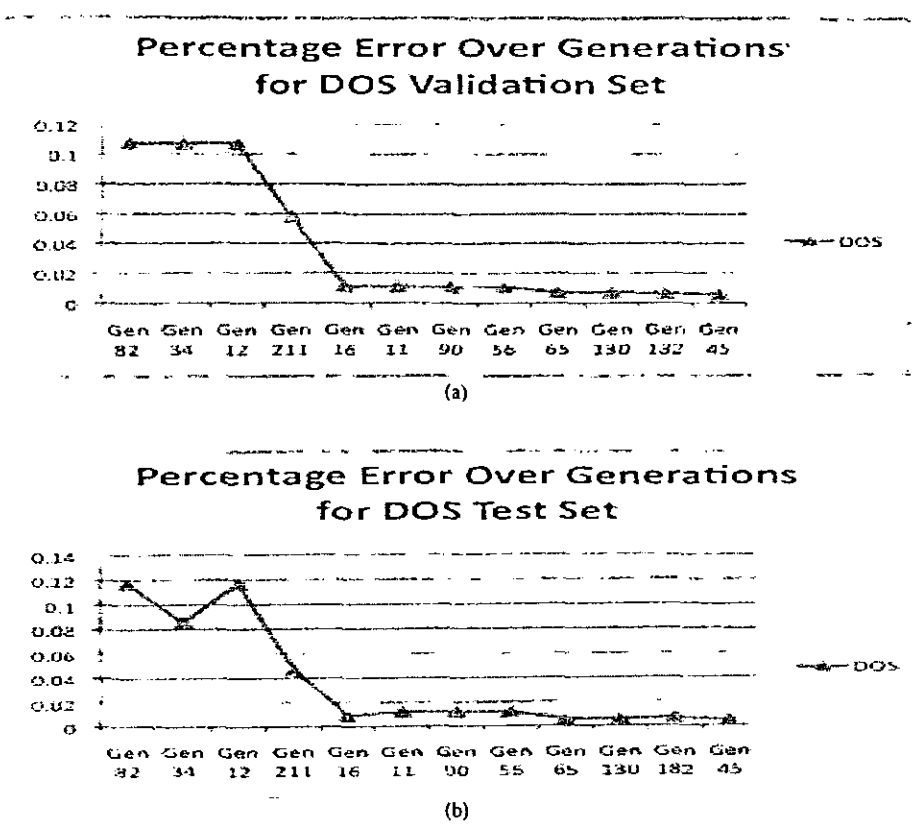


Fig. 6.2. Percentage error (a) validation data (b) testing data for DOS detection.

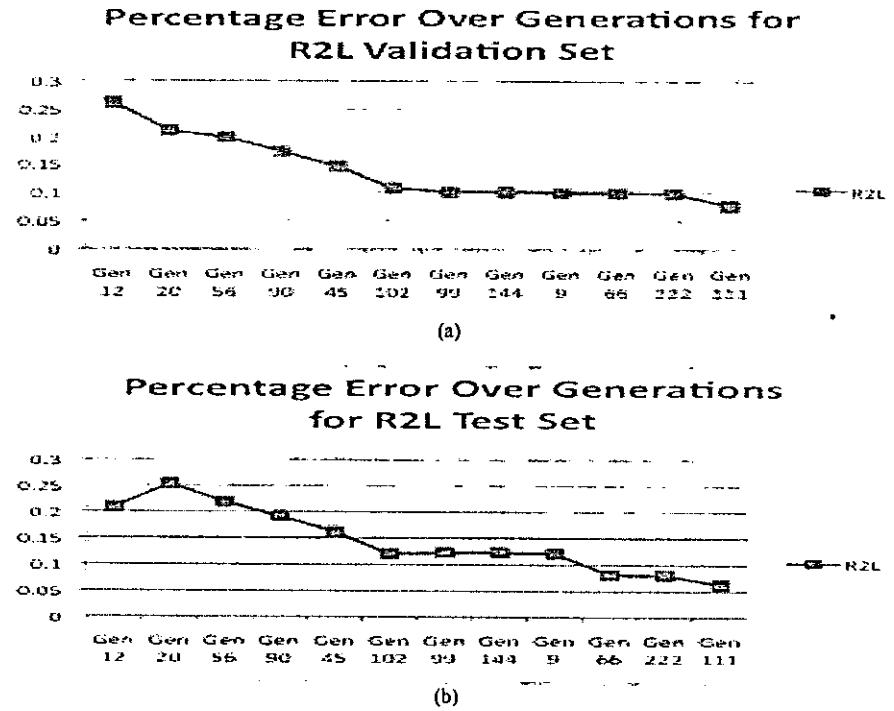


Fig. 6.3. Percentage error (a) validation data (b) testing data for PROBE detection.

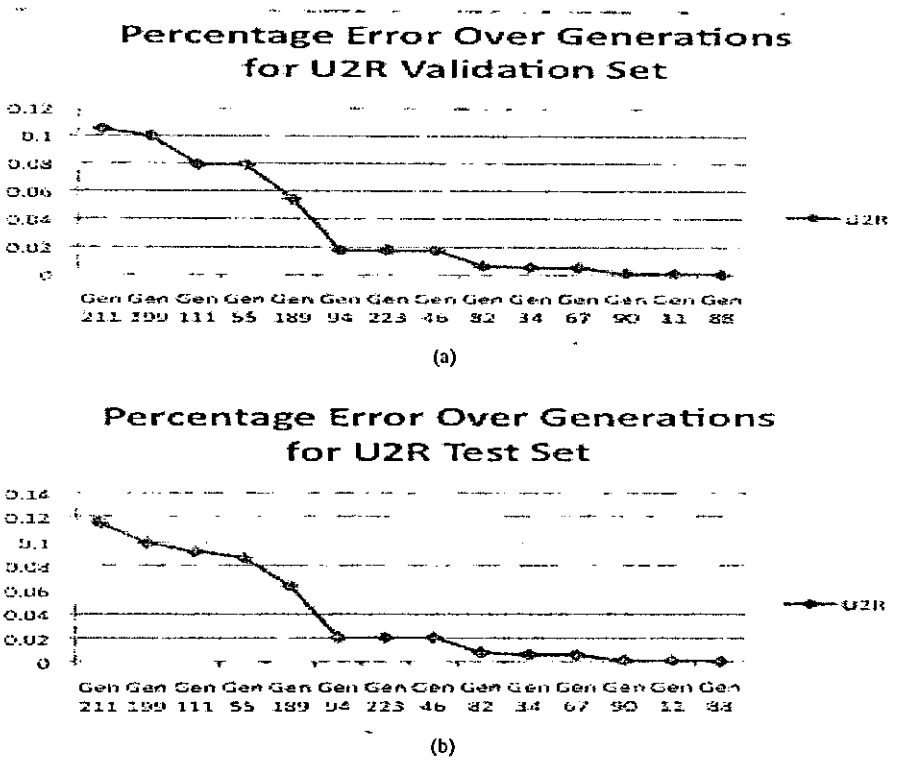


Fig. 6.4. Percentage error (a) validation data (b) testing data for PROBE detection.

6.2 Accuracy for the Classification of Different Intrusions

The accuracy for the classification of different intrusions is shown in Figure 6.5. A maximum accuracy of 99.93% is achieved for U2R intrusions. While a minimum of 93.73% is achieved for R2L attacks. The DOS and PROBE attacks classification achieved an accuracy of 99.53% and 99.33% respectively. This shows the effectiveness of using

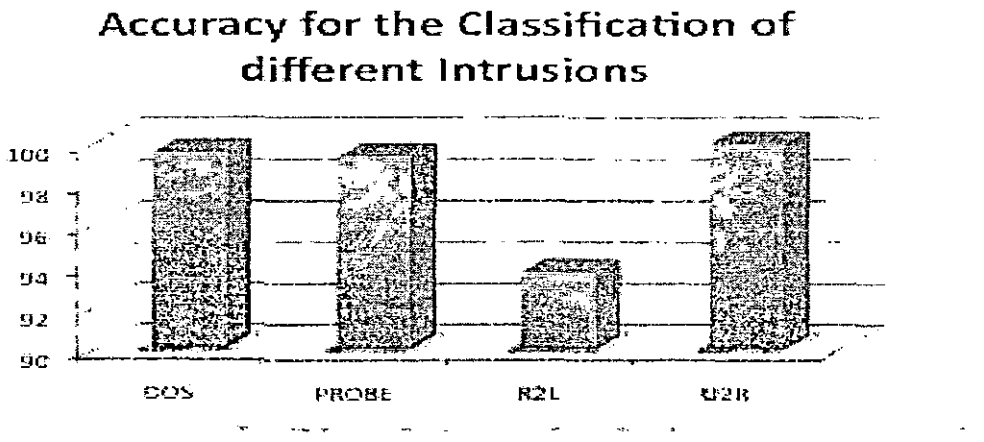


Fig. 6.5 Accuracy for different intrusion attacks

Table 6.1 Comparison of Error Rates with GA based DT and RT [3]

	DOS	PROBE	R2L	U2R
GA based RT	0.004726	0.006626	0.0627219	0.000607
GA based DT [3]	2.321258	2.166494	19.979205	0.095601
Hybrid (Avg) [3]	2.222582	1.670193	19.9545019	0.100885
Hybrid (Best) [3]	2.197655	0.869377	19.628280	0.089016

rough tree instead of typical decision trees, which are based on Genetic algorithms. U2R achieved less classification accuracy as compared with other intrusion because of less

available data set for training and testing purposes. These accuracies are remarkable if we compare the results of this research with previously studied experiments. Table 6.1 summarizes the previously studies on the same topic and the current research. This table clearly indicates that our research achieved better results and minimum percentage of errors are observed using genetic algorithm based rough trees.

6.3 Generation of Trees based on Rough Set Theory

The different rough trees on which maximum accuracies are achieved as discussed in Table 6.1 are shown in Fig. 6.7 – Fig. 6.9. The graph visualization software (graphviz) is used for the generation of different rough trees. Each node indicates a Boolean value depending upon the corresponding feature generated by genetic algorithm. These Boolean values are represented by "yes" or "no" in the tree. The edges between two nodes represent the weight, which is dependent on the causal relationships of previous consecutive nodes.

Finally the results obtained through previously studied research are shown in Fig. 6.6. Which shows that better results can be generated through GA based rough tree.

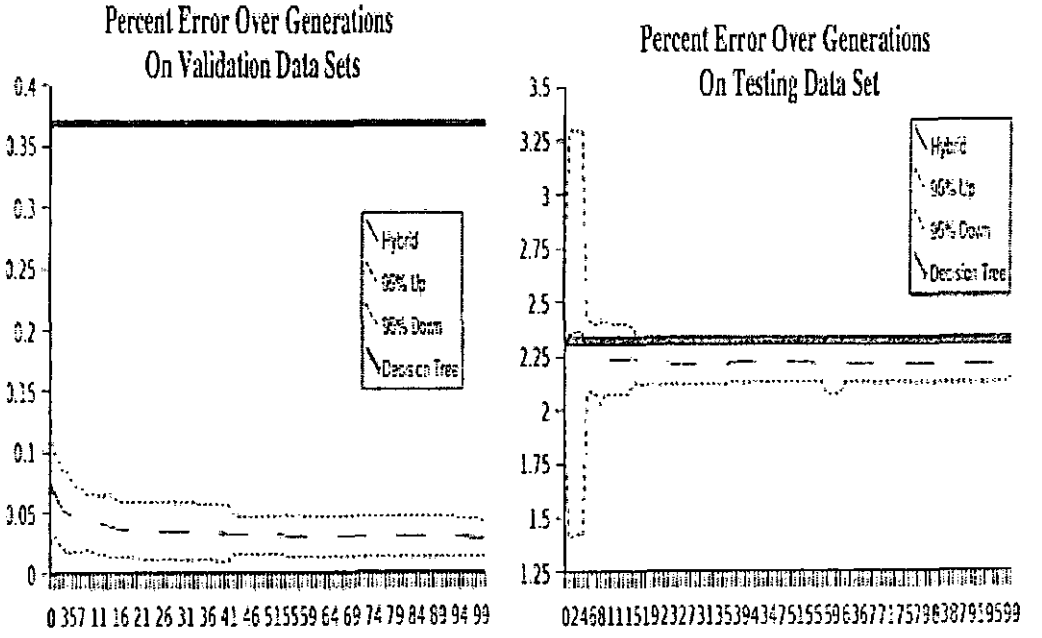


Fig. 6.6 The percentage of error generated in previous research [3]

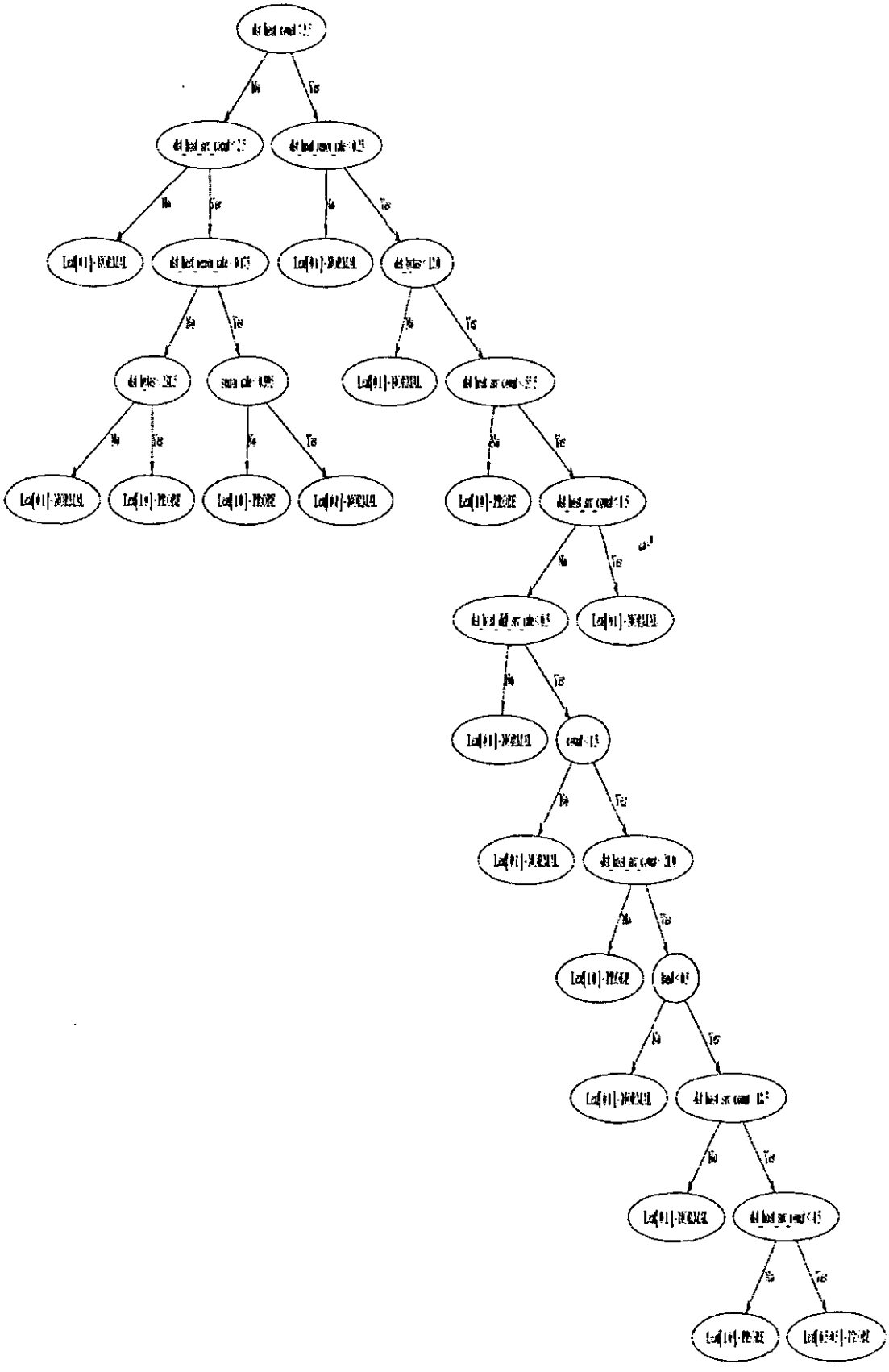


Fig. 6.6 GA based Rough Tree for Dos attack detection.

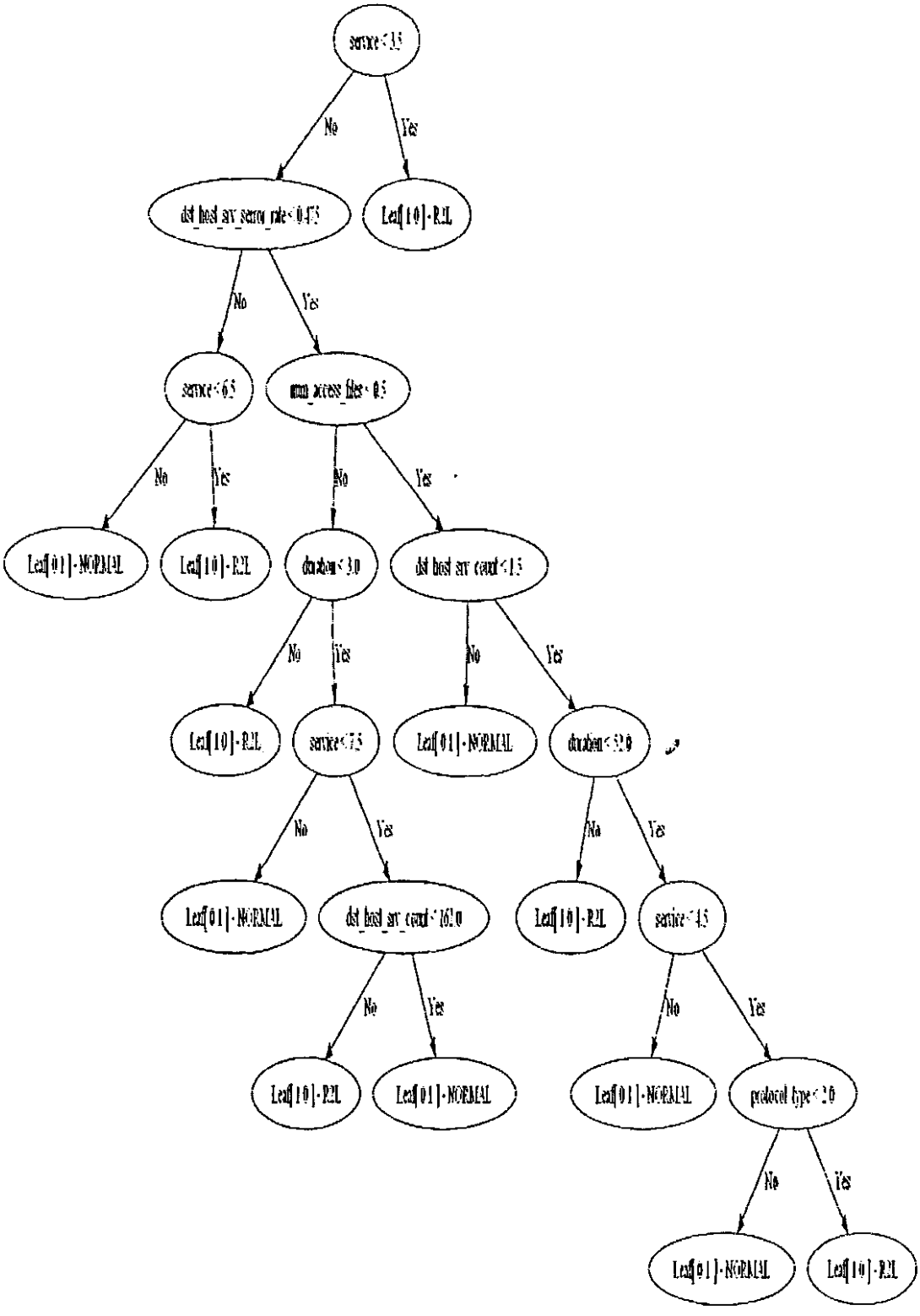


Fig. 6.6 GA based Rough Tree for R2L attack detection.

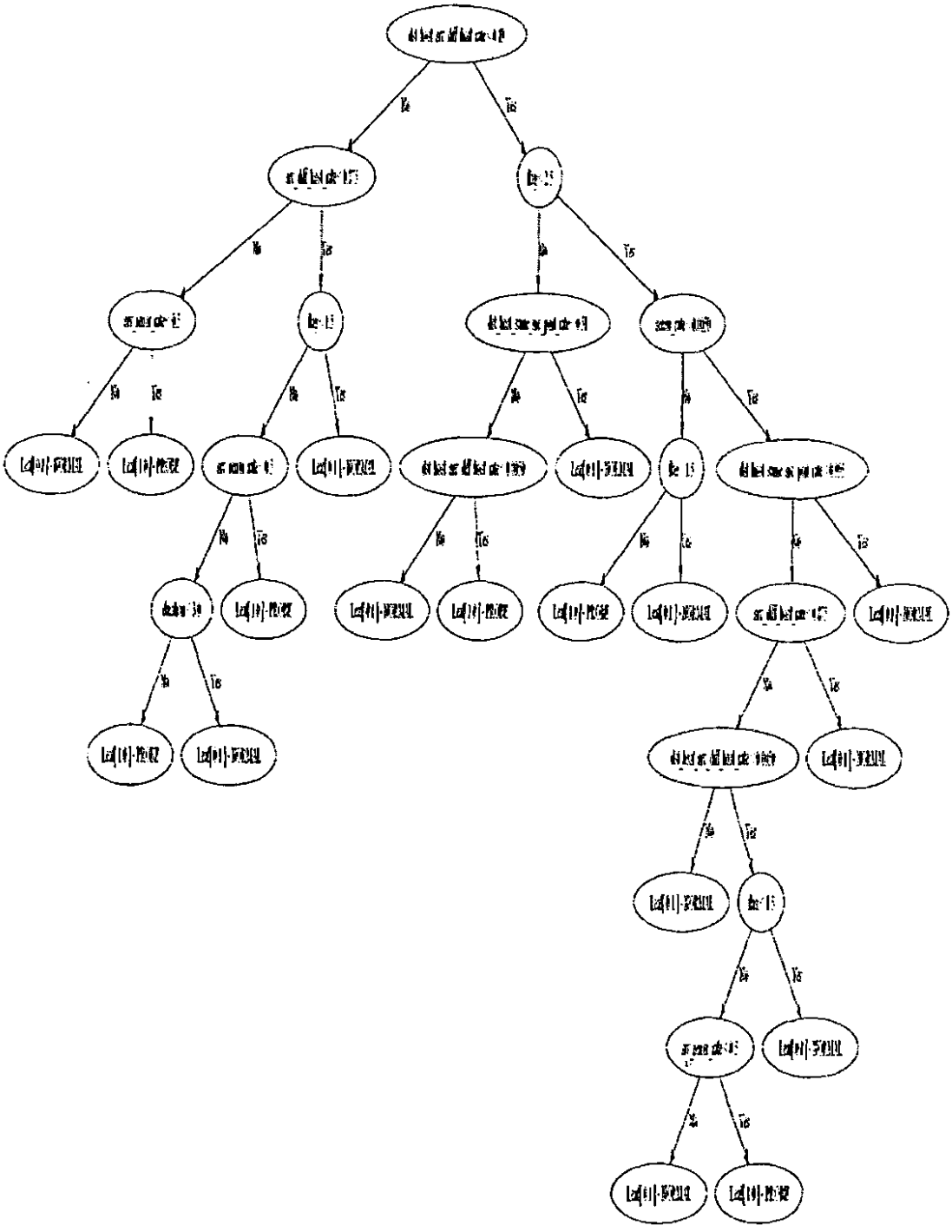


Fig. 6.6 GA based Rough Tree for Probe attack detection.

Chapter 7
Conclusion

7. CONCLUSION

In this chapter, I summarize the thesis and discuss future work.

7.1 Summary

This thesis is based on a hybrid which consists of combination of Rough Trees based on Rough set theory, and Genetic Algorithm. It provides a framework for more accurate Intrusion Detection. There are two modules of this application. Rough Tree is working for the purpose of evaluation, and Genetic algorithm is used as Search component. It works iteratively and gets the generations after generations until gets the best individual. The dataset used is KDDcup99. The information provided by the dataset is passed to Rough Tree Evaluator. The Evaluator provides the performance edge as well as scalability for the further research.

Rough Tree evaluator basically reduces the set of features by skipping the noisy, irrelevant and unimportant features of the data. It outputs only those features which take an important part while taking decision about a connection that either it is an intrusion or a normal connection. The output of this evaluator is shown in the form of Tree. And the rules generated by the Evaluator can easily be learned to detect intruders.

I provided background on intrusion detection techniques, research done on them and also pointed out that current intrusion detection systems lack effectiveness, adaptability and extensibility. I also explained the strengths of my chosen techniques over other techniques.

While working on project, I mainly consider the main need of set of features that are likely to provide high information gain measures in taking decisions. After developing the application, I analyze the outputs by training, validation and testing the data. All experimentation done provides more accurate, concise and intuitive classification rules.

The focus of my thesis is to improve the network intrusion detection process, especially data processing by working with hybrid techniques i.e., Rough Trees and Genetic Algorithm. It can be clearly seen that DOS and U2R are providing minimum error rates. So these categories of intrusion can be found better. The contribution of R2L and U2R in

the dataset used is very low in training data, still DOS and U2R provides more then 99 % accuracy. The accuracy level of R2L can be is comparatively lesser then all. It can be hoped that this application will improve the detection rate of Intrusions especially DOS and R2L and industry as a whole can progress forward more rapidly.

7.2 Future Work

In my thesis, there are a lot of points, which can be considered for further research. After having promising results by using this hybrid technique for intrusion detection, still there exists provision of enhancement.

My algorithm and its results are totally dependent on the dataset used, which is really a rich dataset in actual. I have taken only 10 % of that data and got the enhanced and accurate results. But these results are totally biased. So, in future it is needed to work on full dataset as a whole. As my dataset is not imaginary dataset. Its values are providing real world scenarios. So, more comprehensive results could be found in the follow up of application. This algorithm can also be applied to other datasets to make it generic.

Appendix A: References

[1] Srilatha Chebrolu, Ajith Abraham' and Johnson P. Thomas ;Feature deduction and ensemble design of intrusion detection systems,2004.

[2] Ajith Abraham, Václav Snášel, Pavel Krömer, Sohail Awais; Survey Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques,2008.

[3] Gary Stein, Bing Chen, Annie S. Wu, Kien A. Hua; Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection, 2005.

[4] Joon Hur a, Jong Woo Kim; A hybrid classification method using error pattern modeling, 2008

[5] Athanassios Papagelis, Dimitrios Kalles; GATree: Genetically Evolved Decision Trees,2000

[6] Bai-Ning Jiang1 Xiang-Qian Ding2 Lin-Tao Ma2; A Hybrid Feature Selection Algorithm Combination of Symmetrical Uncertainty and Genetic Algorithm, 2008.

[7] Shiba *0.* A., Saeed W., Sulaiman M. N., Ahmad F. and Mamat A.;Towards An Optimal Feature Subset Selection,2003.

[8] You Chen1,2, Yang Li1,2, Xue-Qi Cheng1, and Li Guo1; Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System,2006

[9] Chris Sinclair, Lyn Pierce, Sara Matzner; An Application of Machine Learning to Network Intrusion Detection, 1999.

[10] Srinivas Mukkamala, Guagalupe, Andrew Sung; Intrusion Detection Using Neural Networks and Support Vector machines, 2002.

[11] Nahla Ben Amor, Salem Benferhat, Zied Elouedi; Naive Bayes vs Decision Trees in Intrusion Detection Systems, 2004.

[12] Sophia Kaplantzis, Nallasamy Mani; A STUDY ON CLASSIFICATION TECHNIQUES FOR NETWORK INTRUSION DETECTION, 2005.

[13] Ron Kohavi; Scaling Up the Accuracy of Naïve-Bayes Classifiers a Decision-Tree Hybrid,1996.

[14] Yangsheng Ji, Lin Shang; RoughTree A Classifier with Naive-Bayes and Rough Sets Hybrid in Decision Tree Representation, 2007.

[15] Andrew H. Sung and Srinivas Mukkamala ;The Feature Selection and Intrusion Detection Problems,2004.

[16] KDDCUP 1999
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html