GENERTIA: A SYSTEM FOR VULNERABILITY ANALYSIS, DESIGN AND

REDESIGN OF IMMUNITY-BASED ANOMALY DETECTION SYSTEM

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

_____

Haiyu Hou

Certificate of Approval:

_____	_____
David Umphress Associate Professor Computer Science and Software Engineering	Gerry Dozier, Chair Associate Professor Computer Science and Software Engineering
_____	_____
Richard Chapman Associate Professor Computer Science and Software Engineering	Joe F. Pittman Interim Dean Graduate School

GENERTIA: A SYSTEM FOR VULNERABILITY ANALYSIS, DESIGN AND REDESIGN OF IMMUNITY-BASED ANOMALY DETECTION SYSTEM

Haiyu Hou

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirement for the

Degree of

Doctor of Philosophy

Auburn, Alabama
December 15, 2006

GENERTIA: A SYSTEM FOR VULNERABILITY ANALYSIS, DESIGN AND REDESIGN OF IMMUNITY-BASED ANOMALY DETECTION SYSTEM

Haiyu Hou

_____
Signature of Author

_____
Date of Graduation

VITA

Haiyu Hou, son of Chengya Hou and Shuquan Deng, was born June 8, 1973, in Panzhihua, Sichuan, China. After graduation from Panzhihua No. 3 High School in 1991, he attended Sichuan University and graduated with a Bachelor of Science degree in Biochemistry in July, 1995. Then he studied Molecular Biology for three years at Shanghai Institute of Biochemistry, Chinese Academy of Sciences. In June, 1998, he entered Auburn University and graduated with a Master of Science degree in Textile Science in May, 2000. In May, 2002, he obtained a second Master of Science degree, in Computer Science. After that, he continued his study in Computer Science for a Doctor of Philosophy degree at Auburn University.

DISSERTATION ABSTRACT

GENERTIA: A SYSTEM FOR VULNERABILITY ANALYSIS, DESIGN AND REDESIGN OF IMMUNITY-BASED ANOMALY DETECTION SYSTEM

Haiyu Hou

Doctor of Philosophy, December 15, 2006
(M.S., Auburn University, 2002)
(M.S., Auburn University, 2000)
(B.S., Sichuan University, 1995)

169 Typed Pages

Directed by Gerry Dozier

The principles of immunology have been applied to the design and implementation of artificial systems and algorithms for solving a broad range of mathematical and engineering problems, which results in a new computation paradigm, termed an Artificial Immune System (AIS). This dissertation focuses on the performance improvement of an AIS in its anomaly detection functionality.

A typical AIS can be described as having three factors: 1) pattern and detector representations, 2) matching rules that decide the affinity between detectors and patterns, and 3) algorithms that describe the generation, death and regeneration of detectors.

Traditional representations and matching rules have been shown to make AIS suffer from a series of problems including poor scalability. This dissertation proposes a constraint-based representation and the corresponding matching rules to address these problems.

This dissertation proposes GENERTIA, a system that proactively improves the performance of an AIS by discovering and patching the vulnerabilities. GENERTIA consists of two subsystems: a red team and a blue team. The red team is able to discover the vulnerabilities in the AIS, and the blue team design detectors to patch the discovered vulnerabilities. The two teams effectively strengthen an AIS in an interactive and co-evolutionary fashion.

GENERTIA is applied to an AIS-based intrusion detection system (IDS). Experiments show that GENERTIA can effectively increase the detection rate of the IDS with little increase of false positive rate.

The GENERTIA blue team provides a novel approach to the generation of a compact and effective detector set. This leads to the proposal of an anomaly detection-based classification system. This classification system consists of multiple subsystems with each subsystem being an AIS that discriminates a class of patterns from other classes.

The GENERTIA is also applied to the proposed classification system to improve its classification accuracy by improving the performance of the individual subsystems of the classification system.

Style manual or journal used: Evolutionary Computation

Computer software used: Microsoft Word 2003

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Biology has always been a rich source of inspiration for developing computational models and problem solving methods [De Castro and Von Zuben 2004]. Artificial Neural Networks [Haykin 1999] and Evolutionary Computation [DeJong and Spears 1993] are good examples. During the past two decades [Dasgupta 1999], the immune system has drawn significant attention, and the theories and metaphors of the immune system have been successfully applied to the development of systems and algorithms for solving scientific and engineering problems. As a result, a new intelligent computational paradigm, Artificial Immune System (AIS), has come into being [De Castro and Timmis 2002a, De Castro and Timmis 2003].

The natural immune system (NIS) is an extremely complex system, and some key immune principles, such as self-nonself discrimination, are still subjects actively debated by immunologists with contradictory theories [Langman and Cohn 2000]. The main purpose of the immune system is to protect the body from various harmful entities, which, in most cases, are from outside of the body, such as bacteria, viruses, fungi and protozoan parasites, which are the major causes for many diseases and ailments, so that they are collectively termed pathogens [Janeway and Travers 2001].

The NIS has complicated self-nonself discrimination mechanisms to effectively detect and eliminate these pathogens. Another prominent feature of the NIS is its ability to remember the pathogens that have been successfully recognized so that the system can launch a rapid and more specific response when the same or similar pathogens reappear. These two immunological mechanisms are the most interesting metaphors for the computer scientists, and accordingly, most AIS research can be classified into two categories [Dasgupta 1999]: 1) Techniques inspired by the self-nonself recognition mechanism and, 2) Techniques inspired by the immune memory mechanism.

The AIS community is interested in the development of novel models and algorithms with inspiration from immunology as well as the application of models and algorithms to new industrial or engineering application areas [Timmis et al. 2004, Hart and Timmis 2005]. A pioneering study by Farmer et al. [Farmer et al. 1986] developed a computational model that is based on the Idiotypic Network Theory, which was proposed by theoretical immunologist, Jerne, [Jerne 1974] to explain the immune memory mechanism. In this work, Farmer et al. [Farmer et al. 1986] revealed the similarity between their model and the classifier system introduced by Holland et al. [Holland and Reitman 1978], which suggested the possibility of developing immunity-based machine learning techniques. Following this novel idea, Cooke and Hunt [Cooke and Hunt 1995, Hunt and Cooke 1995, Hunt and Cooke 1996] proposed a supervised machine learning

technique using an AIS for DNA sequence classification. Based on these studies, Timmis et al. [Timmis et al. 2000] proposed an AIS that can be applied to data analysis and clustering. Following this line of research, Watkins and his colleagues [Watkins and Boggess 2002, Watkins and Timmis 2002, Watkins and Timmis 2003] proposed an efficient supervised learning algorithm for pattern classification.

Research based on self-nonself recognition can be traced back to the pioneering work done by Forrest and her group when they proposed the negative selection algorithm (NSA) [Forrest et al. 1994] that was inspired by the clonal deletion process in the vertebrate thymus to remove self-destructive T-cells to prevent autoimmunity [Forrest et al. 1997, pp. 15-16 in Janeway and Travers 2001]. The NSA is primarily used to generate detectors for anomaly detection. The anomaly detection problem can be defined as follows [Tanase 2002]: given a set of samples with assigned values, establish a function to classify future samples as normal or abnormal. Usually, only normal samples are available (or only normal samples are used) for building the function. From the perspective of the NSA, the universe of patterns is divided into two parts, the self space, which consists of all normal patterns, and the nonself space, which consists of all abnormal patterns. The self and nonself spaces are complementary to each other, and there is no overlap between them. The NSA generates detectors that tend to cover only the nonself space.

Since the introduction of the NSA, much research has been devoted towards improving the efficiency and effectiveness of the NSA [D'haeseleer et al. 1996, Wierzchon 2000, Ayara et al. 2002, Balthrop et al. 2002b, Gonzalez et al. 2002, Singh 2002, Gonzalez et al. 2003, Esponda et al. 2004] and the application of it to a broad range of application areas [Dasgupta 1999]. These include computer security [Forrest et al. 1994, Forrest et al. 1996, Forrest et al. 1997], network intrusion detection [Hofmeyr et al. 1998, Hofmeyr 1999, Hofmeyr and Forrest 2000, Kim and Bentley 2001b, Hou et al. 2002, Harmer et al. 2002, Dasgupta and Gonzalez 2002], industrial mill fault detection [Dasgupta and Forrest 1995, Dasgupta and Forrest 1996, Taylor and Corne 2003, Dasgupta et al. 2004], pattern classification [Lee and Sim 2004], to name a few.

Although previous studies have shown the AIS to be a promising problem solving technique, there is still much work that needs to be done to make the AIS truly useful for real-world applications, with the following being some major research issues: enhancement of the representations, improvement of the efficiency and effectiveness of AISs, exploration of immune mechanisms that are not introduced to current AISs, and integration of existing AIS algorithms for improved performance.

1.1 Goals

This dissertation deals with AISs that are based on a self/nonself recognition mechanism. The main purpose of this dissertation is to contribute some ideas that, hopefully, improve

the performance of the anomaly detection ability of AISs. Specifically, the goal is to make an AIS more effective by improving its ability to correctly distinguish self from nonself. This goal is met through the realization of the following main objectives:

1. To define different encoding schemes for the self/nonself representation, for which binary-string representation is very common. A binary representation makes the analysis of the problem space easier; however, this representation usually does not provide information that can be easily understood by a human. Since the AIS deals with data with rich semantic content, a higher level representation will facilitate the abstraction of meaningful knowledge.

2. To define a new detector representation. Strong detectors are needed to improve the performance of an AIS. With a binary representation, detectors have constant size of detection range. The NSA has been shown to have a severe scalability problem [Kim and Bentley 2001a] so that AISs based on the NSA cannot deal with high-dimensional data sets. This scalability problem may be solved or alleviated with a better designed detector representation. Different detector representations also suggest the definition of different matching schemes.

3. To develop a new detector generation algorithm. The effectiveness of an AIS depends on the size of its detector set. AISs that use the NSA for detector generation are not efficient [Kim and Bentley 2001a, Stibor et al. 2005c]. A more efficient

detector generation algorithm should be able to generate a small detector set with high effectiveness. This is especially necessary for real-time applications such as network intrusion detection, where a large detector set may not be suitable [Kim and Bentley 2001a].

4. To improve the performance of an AIS by reducing the number of 'holes' in the system. In other words, to increase the detection rate of the AIS. Nonself may go undetected through 'holes', which are pieces of nonself space that are not covered by the existing detectors. The existence of holes makes an AIS vulnerable to certain attacks, therefore, holes in an AIS are also referred to as vulnerabilities of the AIS in this dissertation. It is inefficient to reduce the number of holes by increasing the size of detector set. A 'self-inspection' mechanism for discovering and patching holes may effectively improve the performance of the AIS.

5. To verify the techniques proposed in this dissertation, experiments with AIS in two application areas are carried out, including network intrusion detection and pattern classification.

1.2 Main Contributions

The goal of this dissertation is to improve the performance of anomaly detection- based AISs. This is achieved by the introducing of a new self/nonself and detector representations, which are referred to as a constraint-based representation [Hou et al.

2002]; and accordingly, new matching schemes, including an any-*r*-interval matching rule and a contiguous-*r*-interval matching rule, are designed for the new representation. Experiments [Hou and Dozier 2004] have shown that AISs with the new representation have better performance than AISs with the traditional binary representation.

A genetic algorithm-based system, termed GENERTIA [Dozier 2003, Dozier et al. 2004b, Dozier et al. 2004a, Hou and Dozier 2005], is developed to improve the performance of an AIS by iteratively discovering and patching holes in the system. GENERTIA consist of two subsystems: the GENERTIA red team and the GENERTIA blue team. The objective of the GENERTIA red team is to discover the vulnerabilities in the AIS. The objective of the GENERTIA blue team is to design detectors for the vulnerabilities discovered by the GENERTIA red team. The two teams cooperate in an interactive co-evolutionary fashion to strengthen the AIS.

Detectors are designed to cover nonself space. Coverage overlap between detectors is inevitable, but coverage overlap does not provide extra detection ability for a system. AISs that depend on the NSA may generate a large number of overlapping detectors. With GENERTIA, detectors are only generated to cover nonself space that is not covered by existing detectors in the AIS. This effectively reduces the coverage overlap between detectors, which results in an AIS with a compact detector set and high performance [Hou and Dozier 2006b].

AISs designed by GENERTIA have very compact detector sets and high performance [Hou and Dozier 2005]. This provides an opportunity to develop classification systems that are based on anomaly detection [Hou and Dozier 2006a]. An anomaly detection based classification system is proposed. The proposed classification system consists of multiple subsystems, with each subsystem being an AIS for one class of the classification problem. Experimental results on benchmark data sets show that such a classification system is favorably comparable to well-known classifiers, and the compact detector set can be easily converted into rules that are meaningful for humans [Hou and Dozier 2006a].

The performance of the classification system can be further improved by applying GENERTIA to the individual subsystems. GENERTIA can reduce the vulnerabilities (holes) in the subsystems and therefore effectively reduce the number of false negatives that make some patterns unclassifiable.

1.3 Dissertation Outline

Chapter 2 presents a biological background on immunology, and focuses on the basic concepts that are necessary for understanding AIS. Chapter 2 then provides a survey of AIS research that is related to the work proposed in this dissertation.

Chapter 3 presents a constraint-based AIS that performs anomaly detection. Chapter 3 begins with the introduction of the rationale of a simple AIS that uses the NSA to generate detectors, followed by the presentation of the representation of self/nonself in the form of vectors of integers, the constraint-based representation. The any-*r*-interval and contiguous-*r*-interval matching rules are also introduced.

Chapter 4 introduces GENERTIA which consists of two subsystems: a red team and a blue team. The GENERTIA red team is used to discover 'holes' or vulnerabilities in the system; while the GENERTIA blue team is used to design corresponding detectors for the holes discovered by the red team. The GENERTIA can be applied to an anomaly detection AIS to recursively reduce the holes in the system.

Chapter 5 presents an application of GENERTIA to an AIS-based intrusion detection system. Experiments are carried out to show the applicability of GENERTIA to immunity-based intrusion detection systems for performance improvement.

Chapter 6 proposes an anomaly detection based classification system. This classification system consists of multiple subsystems with each subsystem being an anomaly detection-based AIS. These subsystems depend on a GENERTIA blue team to generate negative detectors which can be easily converted into classification rules. Experiments are conducted with benchmark data sets to test the performance of this classifier.

Chapter 7 presents the application of GENERTIA to the proposed classification system for better performance. After the classification system is built with the GENERTIA blue team, the red team can be used to discover the vulnerabilities in the subsystems and these discovered vulnerabilities can be patched by the GENERTIA blue team. This process may improve the accuracy of the classification system by reducing the number of patterns that cannot be classified due to false negatives from subsystems.

Chapter 8 gives a summary of the work proposed in this dissertation and provides several directions for future research.

CHAPTER 2 BACKGROUND

This chapter first briefly introduces a number of concepts in immunology. These concepts are necessary for the understanding of AIS. Following the introductory concepts will be a list of immunological principles that are interesting to computer scientists. Next a survey of the research work on AISs that is related to the work presented in this dissertation is provided.

2.1 Immunology for Computer Scientists

This section provides a succinct introduction to the structure and functionalities of the natural immune system and summarizes the principles that have been inspiration for computer scientist and engineers in the development of computational models and algorithms.

2.1.1 Structure and Function of the Immune System

The nature immune system (NIS) exists in all the vertebrates and serves as a vital system for defending the body from various infections that lead to disease [Janeway and Travers 2001]. The NIS is an extremely complex system with many immune mechanisms

not completely understood. This section provides an abstract view of the system while omitting many details of the specific mechanisms. The purpose of this section is to serve as a reference to subsequent discussion on its artificial counterparts. Detailed reviews on the structural and functional analysis of the NIS may be found in [Janeway and Travers 2001, Krogh 2000].

This introduction is organized into the following four parts:

1. "*Innate and Adaptive Immunity*" introduces the two types of immunities that are either specific or nonspecific to the invading pathogens;

2. "*Development of Immunity*" introduces the development of various cells and molecules involved in the immune response;

3. "*Clonal Deletion and Clonal Selection*" introduces the generation of immune cells that are specific to certain pathogens;

4. "*Elimination of Antigens*" briefly describes how the pathogens are eliminated from the body after they are recognized by the immune system.

2.1.1.1 Innate and Adaptive Immunity

The NIS protects the body from infectious diseases caused by various pathogens [Janeway and Travers 2001, Krogh 2000]. There are four major groups of pathogens,

including virus, bacteria, fungi and protozoan parasites. Pathogens elicit immune responses after they enter the body. Substances that can stimulate specific response of the immune system are referred to as antigens, and certain pathogens display specific antigens (usually some membrane proteins) on its surface.

Generic pathogens that display commonly shared antigens on the surface may be recognized and eliminated by a type of leukocyte (white blood cell) referred to as phagocyte, which is attracted to a site of infection to engulf and digest pathogens. This is called innate immunity.

Specific pathogens invoke an antigen-specific immune response, which involves specific lymphocytes that synthesize antibodies binding to a specific antigen and promote the elimination of the antigen. This is called adaptive immunity. The adaptive immune system responds more quickly and efficiently to a repeat infection, and such immunity is long lasting.

2.1.1.2 Development of Immunity

The NIS is a collection of cells and organs that work together to provide immunity [Janeway 2001, Krogh 2000]. NIS cells wander around in the body to detect infections. NIS organs are where leukocytes mature and where they interact with antigens to become fully active effecter cells and memory cells.

The primary immune organs, including the bone marrow and thymus, are where leukocytes form and mature. Lymphocytes are antigen-specific leukocytes responsible for adaptive immunity. They have membrane receptors called antibodies that bind to antigens. Each lymphocyte recognizes one specific antigen. Lymphocytes specific for many diverse antigens are produced continually even in the absence of antigen exposure.

Secondary lymphoid organs, including the spleen and lymph nodes, are where leukocytes meet with antigens. Some fluid leaves the blood circulation at the capillaries and bathes the tissues, supplying nutrients and washing away waste products. The fluid, called lymph, then collects in the lymphatic vessels and passes through the lymph nodes on its way back to the blood stream. If the tissues are infected, antigens are carried to the lymph nodes where they contact with lymphocytes and initiate adaptive immune responses. When a lymphocyte encounters its specific antigen, it proliferates and differentiates into a clone of effecter cells with the same antigen specificity.

2.1.1.3 Clonal Deletion and Clonal Selection

Lymphocytes, a vital component of the immune system, carry antigen receptors that bind with antigens that are carried by pathogens such as virus, bacteria, fungi and protozoan parasites. The binding between an antigen and a receptor is highly specific, which is determined by the physical and chemical characteristics of the binding surfaces of the two molecules. Each lymphocyte carries copies of a certain antigen receptor whose

antigen-binding specificity is determined by its encoding gene. A unique genetic mechanism that operates during lymphocyte development is capable of generating different variants of antigen receptor genes in the magnitude of 10^{11} [pp. 144 in Janeway and Travers 2001], a repertoire that is large enough to bind virtually any possible antigen, including molecules that come from the body itself. The clonal elimination process removes the lymphocytes that carry potentially self-reactive antigen receptors before they mature and join the immunity forces.

A lymphocyte binds an antigen with specific antibodies, which are encoded in groups of gene segments. As lymphocytes develop in the bone marrow and thymus, they must recombine several of these gene segments to produce a functional antibody. Recombination occurs randomly, so millions of lymphocytes produced daily collectively have millions of different antigen specificities. Since receptor generation is random, cells specific for self-antigens are also produced. Anything binding to immature lymphocytes in the thymus and bone marrow results in their death, which is known as clonal deletion [Janeway 2001, Krogh 2000], and that is why mature lymphocytes do not bind to self-components.

Mature naive lymphocytes are those that have not yet bound to any foreign antigen. They leave the primary lymphoid organs and re-circulate through the secondary lymphoid organs. If a lymphocyte does not bind its specific antigen in a few weeks, it

dies. If it does bind to an antigen, it proliferates into a clone of lymphocytes that differentiate into antigen-eliminating effecter cells. Memory lymphocytes specific for the same antigen are also produced. This process is called clonal selection [Janeway 2001, Krogh 2000]. It explains the specificity of adaptive immunity.

2.1.1.4 Elimination of Antigen

The antibody is the primary weapon for antigen elimination in adaptive immunity. After being activated by antigens (and other signals) lymphocytes become plasma cells that secrete antibodies, which is specific to the activating antigens. Antibodies bind to pathogens to deactivate them. They also coat the pathogens, and with the help of other molecules, generate signals to attract phagocytes to engulf and digest the coated pathogens. Many digestive enzymes are released by phagocytes to decompose pathogens in this process.

2.1.2 Immunological Principles for Design of Computer Systems

Although our understanding of the complex immune system is still at a very preliminary stage, computer scientists have gained many valuable insights that have lead to the design of artificial immune systems (AISs) for a broad range of application areas [Dasgupta 1999]. The following is a list of some of the most important immunological principles

that can be applied in the design of an artificial immune system [Hofmeyr 1999, Gonzalez 2003].

1. Distributed Processing: The immune system consists of millions of immune cells and macromolecules that circulate around the blood and lymph system to detect and respond to immune stimulus. These components interact locally to provide protection in a completely distributed fashion. There is no central control or coordination, therefore no single point of failure in the system.

2. Adaptability and Memory: The immune system can adapt to pathogenic structures by learning to recognize them and keep a memory of them. When a certain pathogen is detected for the first time, a primary response is induced and some lymphocytes that recognize this pathogen are kept as memory cells, which can respond more rapidly when the same type of pathogens are encountered. These memory cells serve as signature-based detectors.

3. Anomaly Detection: For the immune system, there are no explicit definitions of pathogens. Any substance that is not the component of the body itself is potentially harmful to the body. The immune system learns self empirically and develops the ability to detect pathogens including the ones that have never been encountered before. The ability to detect novel pathogens is vital because any organism will always encounter them in its lifetime.

4. Diversity and Robustness: There are different immune components that provide diverse pattern recognition so that the immune system can detect a variety of pathogens. These components are multitudinous and redundant so that the loss of a few of these components will not significantly affect the overall functionality of the immune system.

2.2 Artificial Immune Systems

With traditional computation, computer scientists have found it either difficult or impossible to perform a certain key range of tasks associated with design, decision making, logistics and scheduling, pattern recognition, problem solving and autonomous intelligence [De Castro and Timmis 2003]. Over the years, great progress towards designing software for such tasks has emerged by taking inspiration from a range of natural, mainly biological, systems [De Castro and Von Zuben 2004]. Artificial Neural Networks and Evolutionary Computation are good examples of successful applications of the biological metaphor to the solution of difficult problems [Haykin 1999, DeJong and Spears 1993]. The immune system, possessing very powerful information processing capabilities such as feature extraction, pattern recognition, and adaptive learning and memory, has provided computer scientists rich metaphors for developing computational models and problem solving methods [Dasgupta 1999, De Castro and Timmis 2002]. During the past decade, the principles abstracted from the natural immune system have

been applied to build artificial systems for solving a broad range of computational tasks, resulting in a new research paradigm referred to as an Artificial Immune System (AIS) [Dasgupta 1999, De Castro and Timmis 2002].

The immune system is a very complex system that is still under active research [Langman and Cohn 2000]. There are various theories regarding the structures and mechanisms the immune system employs to fulfill its overall objective, and these theories even have contradictory details [Langman and Cohn 2000]. Accordingly, current AIS research has adopted a number of those theories; however a typical AIS implements only a few [Dasgupta 1999]. Among the various immunological theories, three are of most interest: Immune Network Theory, Clonal Selection Theory, and Negative Selection Theory. The following sections review AIS models and algorithms based on these three theories.

2.2.1 Immune Network Theory

Idiotypic Network Theory was originally proposed by Jerne [Jerne 1974] to describe the immune system as a self-regulatory network of B-cells. In this theory, B-cells may recognize each other through the presence of certain paratopes on the surface of the B-cell. Two B-cells are connected if they have matched paratopes and the strength of the connection is proportional to the extent to which they match each other, termed as affinity. Connected B-cells both stimulate and suppress each other in the expression of

antibodies. The stabilization of the network is obtained at an equilibrium state between the stimulation and suppression.

Varela et al. [Varela et al. 1988] proposed a cognitive model of the immune system with four significant functionalities: 1) the recognition of various molecules, 2) the memory of encountered molecules, 3) the discrimination of self and nonself molecules, and 4) the associative memory to recognize unseen molecules. They also proposed that these functionalities are associated with three immune network properties: structure, dynamics, and meta-dynamics. The recognition of antigens by antibodies and the connections between antibodies construct the structure of the immune network. The dynamics describes the changing of the network structure by updating the strength of connection between antibodies upon the encountering of new molecules. Meta-dynamics refers to the changing of the network structure by adding new connections and deleting outdated connections.

A seminal work that suggested developing immunity-based computational algorithms was by Farmer et al. in [Farmer et al. 1986], where the authors developed a computational model that is based on the Idiotypic Network Theory, and suggested a strong similarity between their model and the classifier system introduced by Holland et al. [Holland and Reitman 1978]. In their model, antibodies and antigens are represented as binary strings. An antibody clones multiple copies of itself upon stimulation. The

number of copies produced in cloning is determined by concentration level which, in turn, is determined by four factors: 1) stimulation by matching antigens, 2) stimulation by neighboring antibodies, 3) suppression by neighboring antibodies, and 4) cell death due to a finite lifespan.

Cooke and Hunt [Cooke and Hunt 1995] proposed using an AIS that is based on the immune network theory to distinguish promoter sequences from other non-promoter DNA sequences. This AIS consists of a network of B-cells that generate antibodies to classify DNA sequences, and each B-cell can be stimulated by DNA sequences (antigens) or other B-cells and suppressed by other B-cells as well.

Hunt et al. [Hunt et al. 1999] developed the Artificial Immune Network (AIN) model, in which the B-cell population is made up of two sub-populations: the initial population and the cloned population. The initial population is generated from a subset of training dataset to create the B-cell network, and the remainders are used as training antigens. During the training process, antigens are randomly chosen and presented to the B-cell network. A binding between an antigen and a B-cell results in the clone and mutation of the B-cell. The mutation yields a diverse set of antibodies that can be used for classification. New B-cells are either integrated into the network at the closest B-cells or discarded if integration failed. For any antigen that cannot be bound by the current

B-cells, a new B-cell is generated using the antigen as a template and is incorporated into the network.

Timmis et al. [Timmis et al. 2000] proposed the Artificial Immune NEtwork (AINE), which has some similarity to AIN. During the training process, B-cells have opportunities to be stimulated by the antigens from a training set, and the strength of the stimulation is determined by the Euclidean distance between the antigen and the B-cell involved in the stimulation. Upon stimulation, a B-cell can clone itself and mutations are introduced to produce B-cells with binding varieties. The training process results in a network of B-cells that represent the structure of the training set and it can be later used for data clustering.

Based on AINE, Timmis and Neal [Timmis and Neal 2000] proposed the Resource limited Artificial Immune Network (RAIN) to address efficiency problems that resulted from the explosion of B-cell population in AINE. In RAIN, an artificial recognition ball (ARB) is used to represent a number of similar B-cells. The concept of ARB is borrowed from the idea of recognition ball in theoretical immunology, which refers to the region in the shape space that an antibody can recognize. Each ARB is assigned a number of B-cells depending on its level of stimulation; therefore, the whole network is resource limited.

Watkins and his colleagues [Watkins and Boggess 2002, Watkins and Timmis 2002, Watkins and Timmis 2003] proposed the Artificial Immune Recognition System (AIRS) as a supervised learning algorithm. This algorithm consists of four stages. The first stage involves data normalization, parameter discovery and the seeding of memory cells and initial ARB population using zero or more randomly selected training samples. In the second stage, a training sample (regarded as an antigen) is presented to the memory cell population and the memory cell with the highest affinity with the antigen is selected for generating ARB. In the case when no matching memory cell is found, a new memory is generated using the antigen as a template and added to the memory cell population. Then, through the cloning and mutation of the selected memory cell, ARBs are produced and added to the ARB population. In the third stage, ARBs compete for resources according to their stimulation levels (the ability to correctly classify a given antigen), with resources allocated from ARBs with lower stimulation levels to ARBs with higher levels. The ARBs with zero resource are removed from the population. In the final stage, for each antigen, the ARB with the highest stimulation level is compared with the memory cell selected in the second stage, and the one (either the ARB or the memory cell) with a higher stimulation level is added to a set of established memory cells. At this point, the next antigen in the training set is selected and the training process proceeds with the second stage. This process continues until all antigens have been presented to the system.

2.2.2 Clonal Selection Theory

According to *The American Heritage Stedman's Medical Dictionary*[1], Clonal Selection Theory is "*the theory that the mutation of stem cells produces all possible templates for antibody production and that exposure to a specific antigen selectively stimulates the proliferation of the cell with the appropriate template to form a clone or colony of specific antibody-forming cells*". Simply put, the Clonal Selection Theory says that an antigen selects from among a variety of lymphocytes those with receptors capable of reacting with part of the antigen. As a result of this interaction, specific lymphocytes are activated to proliferate (clonally expand). They may then secrete molecules of antibody that can combine with the antigen. If the antigen is part of the surface of a virus or bacterium, then the antibody labels that organism as foreign ("non-self"). The organism is then ingested by phagocytic cells and degraded.

De Castro and Von Zuben [De Castro and Von Zuben 2000] developed an artificial immune network model referred to as aiNet, for pattern recognition and clustering. The network cells in aiNet are represented in a similar way as in AINE and a Minimum Spanning Tree algorithm is employed to identify clusters. The input data are assumed to be unlabeled and a competitive learning algorithm is used to iteratively reduce the mean square error of the input data clustering. The recognition of an input

[1] *The American Heritage® Stedman's Medical Dictionary, 2nd Edition © 2004 by Houghton Mifflin Company*.

pattern (antigen) results in the proliferation, mutation and selection as suggested by the Clonal Selection Theory.

Based on the aiNet model, de Castro and Von Zuben [De Castro and Timmis 2002b] proposed a multimodal function optimization algorithm, named CLONALG, for machine learning and pattern recognition. This algorithm has two repertoires of strings: a set of antigens consisting of training data and a set of antibodies consisting of candidate solutions. The antibody set is further divided into two sub-populations: memory and non-memory repertories. The algorithm runs for a predefined number of generations. During each generation, a randomly selected antigen is presented to the antibody population, and *n* antibodies with the highest affinities are allowed to clone themselves with a cloning rate proportional to their affinities: the higher affinity the more clones generated for each of the *n* selected antibodies. These clones are then subjected to mutation with a mutation rate inversely proportional to their affinities: the higher affinity the smaller the mutation rate. Among these antibodies, the one with the highest affinity replaces the corresponding memory antibody in the memory antibody repertories if it has a higher affinity than the memory antibody; and *d* antibodies with the highest affinities replaces *d* non-memory antibodies with the lowest affinities in the non-memory antibody repertoires. CLONALG is a population-based algorithm that involves mechanisms of reproduction, mutation and selection, which is similar to evolutionary computation (EC). The distinct features of CLONALG, however, are: 1) it can reach a diverse set of locally optimal

solutions, and 2) it takes into account the fitness values of individuals (cell affinity) in order to determine the cloning and mutation rates.

2.2.3 Negative Selection Theory

The negative selection algorithm (NSA), an approach to anomaly detection using negative detectors, was originally proposed by Forest et al. [Forrest et al. 1994] to model the clonal deletion process in the natural immune system to prevent autoimmunity [Forrest et al. 1997, pp. 15-16 in Janeway and Travers 2001]. T-cells, a type of lymphocyte, have antigen receptors covering their surface for binding to antigens. The binding is specific: a certain antigen receptor only binds a specific antigen, while all the antigen receptors on a T-cell have the same specificity. During the generation of T-cells, the gene encoding the antigen receptor undergoes a pseudo-random genetic rearrangement process, which means the antigen-binding specificity of the T-cell is random so that it has the potential to bind any antigen including molecules of the body itself. Therefore, before these T-cells are mature, they are censored in a way that all the T-cells that bind to self-component (cells, macromolecules, etc.) are destroyed, and only the T-cells pass the censoring process are allowed to leave the thymus and circulate in the body to perform immunological functions.

The NSA proposed by Forest et al. [Forest et al. 1994] generates detectors using a similar strategy: detectors are generated at random and censored against a given set of

self patterns. All the detectors that detect any self patterns are removed. This results in a collection of detectors that potentially detect any patterns except the self patterns. The effectiveness of the resulting detector set, in terms of detection of nonself patterns, depends on the size of the detector set. The larger the detector set, the higher the possibility a given nonself pattern will be detected by the detector set. This algorithm is more formally described in [Esponda et al. 2004] from an anomaly detection aspect:

1. Define self, *RS* (real-self), as a set of patterns of fixed length *l*, of which only a subset, *S*, is available for learning. The set of all *l*-length patterns is referred to as *U* (universe). The task of anomaly detection is to distinguish the patterns in *RS* from the anomalous patterns in $U - RS$.

2. Candidate detectors are randomly generated and are censored against *S*; surviving detectors are negative detectors with each covering a subset of $U - RS$. Good coverage of $U - RS$ may be obtained if sufficient numbers of negative detectors are generated.

2.2.3.1 NSA Defined Over Hamming Shape-Space

The NSA does not specify the implementation in terms of pattern representation and matching rule. The NSA proposed by Forrest [Forrest et al. 1994] employed binary strings to represent patterns and detectors, and an *r*-contiguous-bit matching rule was

used to determine the match between patterns and detectors. Under the *r*-contiguous-bit matching rule, a pattern is matched by a detector if they have the same bits in at least *r* contiguous positions. This matching rule has been used to model the antigen-receptor binding in immunology [Percus et al. 1993]. Theoretical analysis on the NSA shows that, with binary-coded detectors and *r*-contiguous-bit matching rule, for certain probability of detection, the number of candidate detectors required for censoring to generate certain number of valid detectors, is exponential in the size of self [D'haeseleer et al. 1996, Ayara et al. 2002]. This causes a scalability problem due to the time complexity that is proportional to the number of candidate detectors and the size of self.

The time complexity problem of the NSA motivated the proposal of other algorithms, including the greedy algorithm [D'haeseleer et al. 1996], linear time algorithm [D'haeseleer et al. 1996], binary template algorithm [Wierzchon 2000] and negative selection with mutation [Ayara et al. 2002], all based on binary string representation. Singh [Singh 2002] proposes an extension of the greedy algorithm for a higher alphabet size. These algorithms have time complexities that are linear in the size of the self, but still suffer from a time complexity that is exponential in the *r* value (the length of shortest substrings that define matching) and require space complexities that are also exponential in the *r* value [Ayara et al. 2002]. While high space complexity is not desirable for network intrusion detection applications, a significant drawback of these algorithms is that they are only applicable to *r*-contiguous-bit matching rules.

Nevertheless, this representation has been adopted by Hofmeyr and Forrest [Hofmeyr and Forrest 2000] to implement LISYS, an AIS-based network intrusion detection system (IDS). Inspired by an earlier anomaly detection-based network security system [Mukherjee et al. 1994], Hofmeyr and Forrest defined self patterns as normally occurring TCP/IP connections that are abstracted as data path triples, including the IP addressees of the two hosts, the port number on the server, and the direction of the connection. A data path triple can be represented as a 49-bit sting. With an extended NSA that generates and censors detectors in an online environment with dynamic self, LISYS obtained satisfying results with simulated network traffic.

Kim and Bentley [Kim and Bentley 2001a] argue that in order to detect intrusions in real network traffic, a more complicated representation of self patterns is required. For example, the connection vector described in [Mukherjee et al. 1994] contains 15 fields. They used a pattern representation with higher dimension (33 fields) and higher cardinality alphabet (0 - 9) and reported extreme difficulty in generating detectors using the NSA due to the serious scaling problem. In their experiments, for 20 minutes worth of network traffic data, 20 hours of training time was needed to generate 1000 valid detectors, and these detectors could only detect less than 16% of the nonself patterns. This lead to their conclusion that the NSA cannot handle real network traffic, and they suggested that this algorithm should only work as a filter for invalid detectors. Balthrop and his colleagues [Balthrop et al. 2002a] argue that the difficulty encountered by Kim

and Bentley in [Kim and Bentley 2001a] was caused by a bad choice of pattern representation (large alphabet) and improper settings of parameters (*r* value). They also point out that the NSA and matching rule are separable pieces and that a bad choice of representation and matching rule may result in poor performance of the NSA.

In another paper, Balthrop and his colleagues [Balthrop et al. 2002b] introduce a variant of the *r*-contiguous-bit matching rule termed as *r*-chunk, where detectors are defined by *r* contiguous bits while other bits are simply considered 'do not care'. A pattern matches an *r*-chunk detector if they coincide at the defined *r* contiguous positions. *r*-chunk matching subsumes *r*-contiguous-bit matching in the sense that any *r*-contiguous-bit matching detectors can be represented as a set of *r*-chunk matching detectors. They have shown that *r*-chunk matching rule may improve the performance of NSA. Formal analysis of the *r*-chunk matching rule is given by Esponda et al. in [Esponda et al. 2004], where they provide formula calculating the total number of generable detectors and the number of holes for which no valid detector can be generated, given *r*, *l*, the pattern length, and $|S|$, the size of the self set used for training. Stibor and his colleagues [Stibor et al. 2004] extend *r*-chunking to higher alphabets and propose an algorithm for efficiently generating *r*-chunk detectors over arbitrary alphabet sizes. Their analysis of this algorithm shows that the *r*-chunk matching rule is not suitable for NSA with higher alphabet size since the alphabet size has a strong negative influence on the total number of generable detectors.

In a subsequent study, Stibor and his colleagues [Stibor et al. 2005a] investigate the appropriateness of applying NSA to network intrusion detection. In this work, empirical and theoretical analysis shows that the number of holes increases exponentially if the *r*-chunk length *r* is not close to *l*, the pattern length. However, for all the detector generating algorithms aforementioned, both the time and space complexities increase exponentially in *r*, which means that NSA will be inefficient for large value of *r*. On the other hand, for a network intrusion detection application, complicated patterns are required to characterize the network connections and packets, which may take 30 bytes [Mukherjee et al. 1994]. Combining these two factors, Stibor conclude that NSA is not appropriate for network intrusion detection application.

Fortunately, the conclusion drawn in [Stibor et al. 2005a] is restricted to the NSA defined over Hamming Shape-Space. Hamming Shape-Space defines a universe that consists of all string of fixed length over a finite alphabet. This includes all the representations mentioned above. However, other options exist, for example, real-coded vectors.

2.2.3.2 Real-Valued NSA

Dasgupta and Gonzalez [Dasgupta and Gonzalez 2002] propose a real-valued representation to characterize the self space and generate detectors to cover the complementary self space. In their approach, the pattern space is normalized into an *n*-

dimensional unitary space, $[0, 1]^n$. Self space is represented as hyperspheres, with each defined by a center and a radius. Detectors are represented as hypercubes, defined by lower and upper values on each dimension. Good detectors should have large coverage and small overlap with self space. They propose to use a genetic algorithm to design such detectors.

Gonzalez and Dasgupta [Gonzalez et al. 2002] propose a Real Valued Negative Selection (RNS) algorithm defined over a *n*-dimensional space, $[0, 1]^n$. In this algorithm, a pattern is simply an *n*-dimensional vector that defines its location in the feature space, while detectors are represented as hyperspheres, defined by an n-dimensional vector that corresponds to the center and by a real value that represents its radius. Pattern-detector matching is defined by the Euclidean distance between the pattern location and the center of the detector and by the radius of the detector. If the distance is less than the radius, then the pattern is matched by the detector and classified as nonself. Detectors are randomly generated and relocated to not match self, and the distance between a detector and self is defined by the median distance of k-nearest self points. A later version of RNS [Gonzalez et al. 2003] also defines a self radius for every self point in order to allow close points to be considered as self.

Ji and Dasgupta [Ji and Dasgupta 2004a] propose a real-valued NSA with variable-sized detectors (termed V-Detectors). This algorithm randomly generates a detector center that lies outside self hyperspheres then continuously increases the radius

of the detector until it contacts with self region. This algorithm terminates when a predefined number of detectors are generated or predefined nonself coverage has been reached, which is estimated statistically [Ji and Dasgupta 2005].

The real-valued NSA is proposed to avoid the scalability problem when the algorithm is defined over Hamming shape-space. It does not apply negative selection to censor invalid detectors. If the termination is decided by the number of detectors, the complexity is linear in the size of the self set used for training [Stibor et al. 2005b]; the complexity is not easily computed, however, if the coverage threshold is used as a termination criteria. With each detector having different volume of coverage, real-valued detectors may have fewer holes than using binary-coded detectors, however, it also makes it difficult to conduct a formal analysis of the algorithm as in [Esponda et al. 2004, Forrest et al. 1994]. Therefore, it is really difficult to predict the performance of the algorithm on a given data set. Although some good results are obtained with real-valued NSA for some data sets [Gonzalez et al. 2002, Gonzalez et al. 2003, Ji and Dasgupta 2004a], very bad results are also reported [Stibor et al. 2005b].

One of the most important parameters for a real-valued NSA is the radius of the self hyperspheres, which affects the performance of the algorithm by affecting the setting of the radii of the detectors. In the aforementioned work on real-valued NSA, this parameter is arbitrarily chosen [Gonzalez et al. 2003, Ji and Dasgupta 2004a], however, Stibor et al. [Stibor et al. 2005b] argue that a proper self-radius can only be determined

when anomalous samples are available. While it is usually the case that both normal and abnormal samples are available for classification problems, abnormal samples are usually not available for anomaly detection-based network intrusion detection, and even they are available, intrusion detection systems that depend on abnormal samples do not truly perform anomaly detection but signature-based detection. Since all the previously mentioned work was conducted with low dimensional data set, Stibor et al. [Stibor et al. 2005b] argue that it is not clear how real-valued NSA performs in high dimensional space.

It also should be noticed that real-valued NSA treats each dimension with equal weight, namely, each of the *n* values from all fields contribute the same to the final decision of self/nonself classification. Whether this is a good idea needs further investigation since intuition tells us that it is very possible that certain fields should have more weight than others in the decision process. The idea of weighing fields differently is weakly and implicitly included in *r*-contiguous-bit matching rules in the sense that not all fields are considered for matching.

2.2.3.3 A Third Representation

Harmer and Lamont's CDIS [Harmer et al. 2002] also employs NSA to generate detectors for virus detection and network intrusion detection. Binary strings of different sizes up to 128 bits were tested for virus detection, and for intrusion detection, 320-bit strings are used to represent network packets that comprise 29 data fields. A variety of

matching rules based on similarity measures have been investigated in searching for a good matching rule. The *Rogers and Tanimoto Similarity Measure* [Rogers and Tanimoto 1960] was announced the best pattern matching rule that provides a good compromise between the specificity and generality of the detectors. This similarity measure is calculated as follows [Harmer et al. 2002]:

Define: $X, Y \in \{0, 1\}^n$

$a = \sum f(i)$, for $i = 1..\ n$, where $f(i) = 1$ if $X_i = Y_i = 1$, otherwise $f(i) = 0$;

$b = \sum f(i)$, for $i = 1..\ n$, where $f(i) = 1$ if $X_i = 1$ and $Y_i = 0$, otherwise $f(i) = 0$;

$c = \sum f(i)$, for $i = 1..\ n$, where $f(i) = 1$ if $X_i = 0$ and $Y_i = 1$, otherwise $f(i) = 0$;

$d = \sum f(i)$, for $i = 1..\ n$, where $f(i) = 1$ if $X_i = Y_i = 0$, otherwise $f(i) = 0$;

Similarity between X and Y is: $(a + d) / (a + d + 2 (b + c))$

If the similarity measure is above a predefined threshold, a matching between X and Y occurs. The pattern representation is exactly the same as that from Hamming Shape-Space, however, the matching rules have made this representation a third category. While the prototype successfully detects nonself strings, unfortunately, it also exhibits low efficiency: the extremely long negative selection time consumed for generating detectors makes it infeasible for real world application.

2.2.4 AIS-Based IDSs

Because of the functional similarity between the immune system and network intrusion detection systems, significant research [Hofmeyr et al. 1998, Hofmeyr 1999, Hofmeyr and Forrest 2000, Kim and Bentley 2001b, Dasgupta and Gonzalez 2002, Harmer et al. 2002] has been done in the past decade in an effort to build immunity-based IDSs.

2.2.4.1 The Need of Anomaly Detection-Based IDSs

With the world becoming increasingly interconnected by computer networks, IDSs have been playing an important role in automatic response to a wide variety of network-borne security problems [Anderson et al. 1995, Kabay 2001]. Currently, most commercial IDSs, such as Cisco Secure[2], Dragon[3], NFR[4] and Snort[5], are signature-based (rule-based) systems. These systems operate by collecting and scanning for signatures of known viruses and attacks. However, this 'reactive' approach makes the network system vulnerable to novel attacks. Updating the number of signatures quickly becomes a burden for system administrators [Leon 2000]. This situation merits the development of 'proactive' systems that are capable of anomaly detection [Hofmeyr and Forrest 2000, Tanase 2002]. In anomaly detection, a model (or profile) of the normal behavior of the system is built, and any discrepancy from the model is deemed as an abnormality. This

[2] Cisco Secure, http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/.
[3] Enterasys Dragon, http://www.enterasys.com/products/ids/.
[4] NFR, http://www.nfr.com/solutions/.
[5] Snort, http://www.snort.org.

approach is based on the assumption that the normal behavior of a system exhibits stable patterns over time. Any observable system behavior can be used to build a model, such as sequences of system calls [Forrest et al. 1996, Hofmeyr et al. 1998], network traffic density [Dasgupta and Gonzalez 2002], and network packet characteristics [Heberlein et al. 1990, Hofmeyr and Forrest 2000, Harmer et al. 2002].

2.2.4.2 Overview of Immunity-Based Intrusion Detection

In the context of anomaly detection, network intrusion detection can be viewed as the problem of classifying network traffic as normal or abnormal, which is comparable to the task of the natural immune system (NIS)[6]. The IS protects the body by detecting and eliminating foreign (nonself) substances such as viruses, bacteria, fungi, and protozoan parasites [Dasgupta 1999]. The similarity between the problem of network security and that faced by the IS has inspired computer scientists to develop artificial immune systems (AISs) for computer security [Forrest et al. 1994, Forrest et al. 1996, Dasgupta 1999, Hofmeyr and Forrest 2000, Esponda et al. 2004].

The research in the area of immunity-based IDSs was pioneered by Forrest and her group in 1994 when thy proposed a change detection algorithm inspired by the self/nonself discrimination in the NIS [Forrest et al. 1994]. The NIS performs anomaly

[6] An alternative theory, referred to as Danger Theory, argues that the immune system responds to danger signals rather than non-self substances. Application of Danger Theory to AISs is a new research branch in the AIS community [15].

detection using a negative detection approach [Janeway and Travers 2001, Esponda et al. 2004]: it produces censored detectors (lymphocytes and antibodies) that potentially bind to everything except self substances, and everything bound by detectors is subject to elimination from the body [Forrest et al. 1997]. A typical AIS-based IDS tries to mimic the natural IS by evolving a population of negative detectors [Hofmeyr and Forrest 2000, Dasgupta and Gonzalez 2002, Harmer et al. 2002, Hofmeyr 1999, Hofmeyr and Forrest 2000, Hou et al. 2002, Hou and Dozier 2004, Kim and Bentley 2001b, Dozier et al. 2004a]: network traffic that is matched by the detectors is classified as abnormal (nonself) and unmatched traffic is classified as normal (self). In [Forrest et al. 1997], Forrest provides an excellent review of the immunological principles that are of interest to the design of AISs.

Detectors of AISs are typically generated using the Negative Selection Algorithm (NSA) [Forrest et al. 1994, Hofmeyr 1999]. In the NSA, candidate detectors are randomly generated, and normal samples are presented to the candidate detectors. The detectors matching any normal samples are replaced by another randomly generated candidate detector. Detectors that do not match any normal samples are considered valid detectors. This process continues until a desired number of valid detectors are generated. This method is efficient in that it employs a relatively small set of detectors that has a high probability of detecting anomalies in the protected system. This is because the detectors have highly variant matching specificities. The assumption behind the

algorithm is that the normal behavior of a system exhibits stable and observable patterns over time, and any perturbation or malfunction in the system will show patterns that are significantly different from the ones observed during the system's normal state.

2.2.4.3 Prototypes of AIS-Based IDSs

In this section, we review some previous research in the field of AIS-based IDSs. We then summarize the advantages and disadvantages of using AIS-based IDSs. More details will be devoted to Hofmeyr's LISYS since the AIS in this dissertation is largely inspired by this system. The structure and implementation of our AIS is given in the next section.

LISYS is an AIS-based IDS [Hofmeyr and Forrest 2000]. It attempts to classify network traffic as either normal (self) or abnormal (nonself). With LISYS, the observable behavior of a network system is modeled using data path triples (DPTs) that represent a TCP/IP connection: the IP addresses of the two hosts involved in the connection, and the service (port) number of the connection. The presumption is that abnormal activities on the network will produce abnormal DPTs that can be distinguished from the DPTs produced from normal activities. The normal DPTs are used to model the normal behavior of the network system, while detectors that detect abnormal behavior are generated in the complement space. Detectors and DPTs are implemented as 49-bit strings and an *r*-contiguous-bit matching rule is used to define the match between a

detector and a DPT. With this rule, two strings match if they have identical bits in at least r contiguous positions. To generate detectors that are adaptable to the normal behavior of the monitored system, LISYS maintains and evolves a population of detectors using an extended negative selection algorithm. The evolutionary process of these detectors is as follows.

Initially, for each host, a randomly generated set of immature detectors is created. These detectors are exposed to normal network traffic (self) for a user-specified amount of time, termed as tolerisation period. During this period, if an immature detector matches a self packet then the detector dies and is regenerated. Immature detectors that have survived the process become mature detectors. Thus, mature detectors will typically match nonself packets. A mature detector has a user-specified lifespan, t_{mature}, to match m_{mature} (termed as match threshold) packets. This time represents the learning phase of a detector. Mature detectors that fail to match m_{mature} packets during their learning phase die and are regenerated as immature detectors. When a mature detector matches m_{mature} packets during its learning phase, a primary response (alarm) is invoked. The match threshold is set to reduce false positives (false alarms), and it is based on the assumptions that nonself packets tend to occur in temporal clumps.

Once a mature detector sounds an alarm, it awaits a response from the network administrator to verify that the detector has identified a valid attack on the system. This response is referred to as co-stimulation [Hofmeyr and Forrest 2000]. If co-stimulation does not occur within a prescribed amount of time the mature detector will die and be regenerated as an immature detector. Those detectors that receive co-stimulation are promoted to being memory detectors and are assigned a longer life time of t_{memory}. Co-stimulation prevents mature detectors from becoming memory detectors by matching self packets. This process models the activation mechanism of mature T-cells, which, besides binding antigen, need a second signal that indicates the occurrence of body damage [Hofmeyr and Forrest 2000]. Memory detectors invoke stronger (secondary) responses when they match at least m_{memory} nonself packets (where usually $m_{mature} > m_{memory}$).

The different finite lifetimes and the dynamics of the three types of detectors renders a rolling coverage of the detector set, which makes LISYS an adaptive system that constantly adjusts its matching behavior according to the current normal behavior of the system. LISYS also uses several other mechanisms to improve the system effectiveness. For example, to reduce holes (nonself for which detectors cannot be generated [D'haeseleer et al. 1996]) in the system, LISYS uses multiple representations (permutation masks); to improve the detection rate on coordinated distributed attack,

LISYS employs distributed detectors and adjusts match threshold using sensitivity level and decay rate.

Many other researchers also have designed immunity-based IDSs using novel algorithms and techniques [Kim and Bentley 2001b, Dasgupta and Gonzalez 2002, Harmer et al. 2002]. Kim and Bentley [Kim and Bentley 2001b] introduced the dynamic clonal selection algorithm (DynamiCS) to deal with dynamic self in a continuously changing environment. DynamiCS is based the dynamics of three types of detectors: immature, mature and memory detectors, which was originally introduced in Hofmeyr's LISYS. In order to reduce the false positive rate, many other mechanisms introduced in LISYS were also implemented in DynamiCS, such as detector tolerisation, activation threshold, and costimulation. DynamiCS is designed to learn by examining a small self sample set at a period of time and replacing detectors that do not represent the current normal behavior. To adapt to the ever-changing normal behavior of a system, DynamiCS introduced the concept of a gene library evolution using hyper-mutation, where immature detectors are generated through mutation on the deleted memory detectors, and this helps the AIS to be able to detect nonself at a satisfactory level without losing self-tolerance.

Dasgupta and Gonzalez [Dasgupta and Gonzalez 2002] present an immunity-based technique that can be used to build a system which generates negative detectors using a genetic algorithm. The detectors can differentiate varying degrees of abnormality

in network traffic. The self and nonself space is viewed as an *n*-dimension space, with each feature of the monitored behavior being a dimension. Normality is defined by the coverage of multiple hyperspheres with a center at each normal sample and a radius *v* that represents allowed variability for generalization. Abnormality in nonself space is viewed as deviation from the normality in self space. In their implementation, the detectors are real-valued and in the form of hyper-cubic rules: for each feature of the monitored behavior, a low/high boundary is used to define the nonself space in the corresponding dimension. This gives a crisp nonself characterization function. To achieve a non-crisp characterization function, different variability values are used to represent results in a collection of detectors hierarchically organized according to the *v* values, and unknown samples can be characterized as the level of deviation from the normality. Dasgupta and Gonzalez also proposed the use of a genetic algorithm that evolves hyper-cubic rules that have large coverage of nonself space and small overlapping with other rules.

Harmer and Lamont's computer defense immune system (CDIS) [Harmer et al. 2002] is a distributed agent-based system built upon a hierarchical layered architecture, capable of both virus and intrusion detection. In this system, network intrusion detectors are built in the form of 320-bit signature strings to monitor packets of three common protocols: TCP, UDP and ICMP. For TCP packets, 29 data fields comprise a detector; for UDP and ICMP, 17 fields are used. Each field takes a value with a range from 1 to 32 bits, and 28 extra bits are used to define whether a particular field is considered valid or

not (first field that defines the protocol type is considered always valid). They proposed that the Rogers and Tanimoto Similarity Measure [Rogers and Tanimoto 1960] was the best pattern matching rule that provides a good compromise between the specificity and generality of the detectors. The detectors are randomly generated and censored using the negative selection algorithm before they are used to monitor the traffic packets. The detectors have a life cycle that is similar to the one proposed by Hofmeyr. When any detector matches a pattern, costimulation from a human expert is required before the detector signals other agent effectors and obtains an extended lifetime. This makes CDIS an interactive system.

There are a number of advantages offered by AIS-based IDSs as summarized in [Hofmeyr 1999, Hofmeyr and Forrest 2000]. AISs designed using immunological principles are robust because they perform distributed and parallel protection, which also makes them error-tolerant. They are self-organized and provide diverse protection in the form of anomaly detection as well as signature-based detection. These systems usually can be tunable to balance between the required resource and effectiveness and between the false positive and false negative rates. One major advantage is that they provide a form of passively proactive protection via negative selection. This enables them to detect novel attacks. Another major advantage is that AISs are capable of adapting to dynamically changing environments. As the characteristics of self traffic change over time, a properly tuned AIS will be able to effectively adapt to the dynamically changing

definition of self. Finally, the detectors evolved by AIS-based IDSs can easily be converted into Snort rules or *TCPdump* filters. This is especially true when constraint-based detectors are used [Hou et al. 2002, Hou and Dozier 2006a]. These constraint-based detectors are easy to understand and can provide network administrators with important forensic information when new attacks are detected.

A significant problem with AIS-based IDSs such as LISYS is scalability [Kim and Bentley 2001a, Stibor et al. 2005b]. It was observed that for systems that are based on the negative selection algorithm (NSA), to maintain certain detection level, the computation time needed to generate detectors increases exponentially with the size of the self set [D'haeseleer et al. 1996, Kim and Bentley 2001a]. Some efficient algorithms have been proposed to alleviate the problem, such as the Linear Time Detector Generating Algorithm and the Greedy Detector Generating Algorithm [D'haeseleer et al. 1996, Ayara et al. 2002]. These algorithms are designed for generating binary-coded detectors that employ an *r*-contiguous-bit matching rule, and they cannot be applied to other detector representations and matching rules. Although these algorithms have a linear time complexity in the size of the self set, they require time and space exponential in *r*, the length of continuous bits that decide a match.

Recently, the appropriateness of applying the NSA to the network intrusion detection has been critically examined in the aspects of efficiency [Kim and Bentley

2001a, Stibor et al. 2004, Stibor et al. 2005a, Stibor et al. 2005b] and effectiveness [Kim and Bentley 2001a, Stibor et al. 2004, Stibor et al. 2005a, Stibor et al. 2005b, Stibor et al. 2005c]. These researches suggest that, if defined over certain representations, the AISs based on the NSA alone could not be effectively and efficiently applied to network intrusion detection. In this work, we use constraint-based detectors, which are real-coded and have varying-sized coverage, to build compact AISs without sacrificing good detection rate.

Another major disadvantage with the use of AIS-based IDSs is that, at present, there is no way to know exactly what types of attacks will pass through undetected. A number of techniques have been developed to reduce the size of potential holes. Hofmeyr and Forrest [Hofmeyr and Forrest 2000] use a permutation mask to introduce diversity in the pattern representation to reduce undetectable patterns, with the assumption that the union of different representations will detect more patterns than any single representation. Balthrop et al. [Balthrop et al. 2002b] propose the concepts of *r*-chunks and crossover closure to achieve greater detector coverage. Dasgupta and Gonzalez [Dasgupta and Gonzalez 2002] use multi-level detectors representing different deviation levels in an effort to model the non-crisp boundary between self and nonself.

Nevertheless, none of these can be used to alert the designer or users as to the types of attacks that will go undetected by the AIS.

CHAPTER 3 A CONSTRAINT-BASED AIS

This chapter presents a simple AIS that is inspired by the self-nonself recognition mechanism of the immune system. It employs the Negative Selection Algorithm proposed by Forest et al. [Forest et al. 1994] to generate negative detectors that can detect anomalies that deviate from the normal behavior or patterns of the system being monitored. This AIS is a simplified version of Hofmeyr's LISYS [Hofmeyr and Forest 2000] with a novel detector representation, which we referred to as constraint-based detectors [Hou et al. 2002, Hou and Dozier 2004]. Timmis and his colleagues proposed that the structure of an AIS can be described with three components [De Castro and Timmis 2002a, Stepney et al. 2004]: the representation of the samples in the self/nonself space, the representation of the detectors, and the matching rule that is used to decide the match between a sample and a detector. We present the constraint-based AIS with these three aspects.

3.1 Pattern Representation

Our patterns are represented in the form of vector of integers, $(x_0, x_1, x_2 \ldots x_n)$, where n is the number of fields that are included in the pattern representation, and x_i is the value of the corresponding field. The cardinality (and range) on each field can be decided by

the actual values on that field in the data set, and for continuous values, it can be decided after discretization as performed in [Kim and Bentley 2001a]. This representation makes the pattern space similar to that defined over Hamming shape-space with higher alphabet cardinality as describe in previous section, however, the detector representation and matching methods make it totally different.

3.2 Detector Representation

Our detectors are implemented in the form of vector of intervals, ($lb_0..\ ub_0$, $lb_1..\ ub_1$, $lb_2..ub_2$ … $lb_n..ub_n$), where each interval corresponds to a field that is included in the pattern representation, and each interval covers the values that fall into that interval [Hou et al. 2002, Hou and Dozier 2004]. We have designed two types of interval that have different coverage shapes. The first one is termed linear interval, in which the lower bound is less than or equal to the upper bound, and the coverage is strictly from the lower bound to the upper bound. The second on is termed circular interval, where the intervals can wrap around so that the coverage is continuous. In this case, the lower bound is not necessary less than the upper bound. For example, an circular interval of 4..1 actually covers 2 sections: 4..*n* and 0..1, where *n* is the cardinality of the alphabet. Notice that circular intervals contain linear intervals.

These two types of intervals are designed for data with and without 'polarity'. We use the term "polarity" to describe the distribution of samples in pattern space. For any field, the max and min values in that field can be deemed as the two poles, and if there is a tendency that values of certain characteristic occur in the vicinity of one pole while values of another characteristic occur in the vicinity of the other pole, we say that this field has polarity. For example, there are 9 fields in the Wisconsin Breast Cancer Data [39], and for each field, there are 10 values from 1 to 10, with higher values indicating a higher likelihood of presence of cancer and lower values indicating a higher likelihood of absence of cancer. Obviously, this data set has polarity. For data sets with polarity, it may be a good choice to employ linear intervals; however, it may be reasonable to employ circular intervals for data sets where a clear polarity is not obvious.

Notice that detectors in this form can be viewed as constraints: a detector population can be viewed as a population of constraints where self corresponds to a set of all solutions that satisfies the constraints and where non-self corresponds to the set of all solutions that violate at least one constraint. We thus refer to this type of detector as a constraint-based detector.

3.3 Matching Rules

We have designed two matching rules for our representation to determine a match between a pattern and a constraint-based detector. An *any-r-interval* matching rule is

similar to rules based on Hamming distance [Hou et al. 2002, Hou and Dozier 2004]. That is, if any *r* numbers of a pattern fall within the corresponding *r* intervals of a detector then that detector is said to match the pattern. For example, given an alphabet of 0…9 with cardinality of 10, a pattern (4, 4, 0, 2, 2, 0) is matched by a circular detector (8..3, 1..1, 4..2, 1..3, 7..8, 0..9) if *r* is not greater than 3. The setting of *r* defines the specificity or generality of the match between a detector and a pattern. An *r-contiguous-intervals* matching rule is similar to the rcb rule: if at least *r* contiguous numbers of a pattern fall within the corresponding *r* contiguous intervals of a detector then that detector is said to match the pattern. In both matching rules, the value *r* works as a matching threshold, defining the specificity or generality of the detectors. The higher the *r* value, the more specific the match between detectors and patterns, and the fewer patterns a detector can match.

Our AIS is a real-coded implementation, which has a significant advantage over the more commonly used binary-coded implementation as described in [Gonzalez et al. 2002]:

> "*The real valued representation is closer to the problem space. In most cases, it is possible to map the detectors back to the original space, giving them a meaning in the problem domain context.* "

In binary-coded implementations (where *r*-contiguous bit matching rule is common), when a detector matches some packet, it will be difficult to analyze how the detector matches it and why the matched packet is abnormal.

Notice that, similar to the real-valued detectors with varied radii, each constraint-based detector may have a different volume of coverage in pattern space, while detector defined on Hamming shape-space has exactly the same volume of coverage. The number of patterns a constraint-based detector can match is determined by the width of each interval. Using detectors with variable coverage to model nonself space has some advantages, for example, it is more economical to use fewer 'large' detectors to cover large continuous nonself space, while it is more convenient to use 'small' detectors to cover small nonself space surrounded by self space, where 'normal-sized' detectors do not fit in. This is exactly how real-valued detectors with variable coverage can avoid 'holes' introduced by using binary-coded detectors with a constant size of coverage.

CHAPTER 4 GENERTIA – A SYSTEM FOR AIS VULNERABILITY ANALYSIS AND REDESIGN

4.1 Introduction

The mission of AISs for anomaly detection can be thought of as learning to distinguish between self and nonself patterns. There are two types of possible errors associated with the anomaly detection process: self (normal) patterns erroneously detected as nonself (abnormal) patterns, and nonself (abnormal) patterns erroneously regarded as nonself (normal) patterns so they are not detected by the system. In the context of statistics, these two types of errors are termed as false negatives and false positives [Marchette 2001]. Accordingly, the performance of these AISs can be evaluated in terms of two measurements: the false negative rate and the false positive rate. The false negative rate, which is the proportion of the nonself that has not been successfully detected by the system, and the false positive rate is the proportion of the self erroneously detected by the system. The false negative rate can also be expressed as 1 minus the corresponding detection rate which is proportion of the nonself successfully detected by the system. False negatives are also referred to as vulnerabilities (holes) [Hofmeyr and Forrest 2000,

Marchette 2001, Harmer et al. 2002, Dozier 2003], in a sense that attacks will go undetected through these holes(in the abnormal pattern space) because the system lacks of detectors that cover the corresponding pattern spaces where these abnormal patterns exist.

The existence of holes may cause nonself patterns go undetected. As mentioned in Chapter 2, one of the major disadvantages of using an AIS is that there is no way to know exactly what nonself patterns will not be detected by the system. A number of techniques have been proposed to reduce the number of potential holes [Hofmeyr and Forrest 2000, Balthrop et al. 2002b, Dasgupta and Gonzalez 2002, Stibor et al. 2004]; however, the problem of possible holes still exist and there is no way to ensure that certain level of confidence has been achieved regarding the possibility of unknown nonself being detected [Hofmeyr 1999].

Dozier et al. proposed a technique to improve the design of AIS-based Intrusion Detection Systems by searching for vulnerabilities in an AIS [Dozier 2003, Dozier et al. 2004a-b]. Based on this fundamental work, we present a system that can be used to improve the performance of immunity-based systems through vulnerability analysis and AIS redesign. This system, which we refer to as GENERTIA (GENetic and EvolutionaRy Teams for Interactive design and Analysis), works by locating and patching the holes in an AIS.

This system consists of two subsystems, the GENERTIA Red Team (GRT) and the GENERTIA Blue Team (GBT). The GRT is for AIS vulnerability analysis, namely, to discover vulnerabilities (holes) in the AIS, while the GBT is for AIS redesign by generating detectors that patch the vulnerabilities discovered by the GRT. The GRT and the GBT cooperate to recursively reduce the holes in the AIS.

The GENERTIA approach may be more efficient in the terms of the additional coverage obtained by the AIS through increasing detectors compared to adding randomly generated detectors through the NSA. This approach also may be more effective in terms of reducing the false positive rate.

4.2 GENERTIA Red Team

The objective of the GRT is to perform vulnerability analysis on the AIS. The concept of the GENERTIA red team was originally proposed by Dozier in [Dozier 2003, Dozier et al.2004a-b] where genetic and particle swarm red teams were used to analyze the vulnerability in an AIS.

In this dissertation, the GRT employs a steady-state Genetic Algorithm (GA) [DeJong and Spears 1993], although other EC paradigms such as Particle Swarm Optimization are also good candidates [Dozier et al. 2004b]. The fundamental idea of the red team is very simple: present the AIS with nonself patterns and let the AIS tell us the

percentage of the detector set that fails to match these patterns. These nonself ('red') patterns can be randomly generated; however, applying a GA makes the discovery of vulnerabilities more efficient. Information about the discovered vulnerabilities can then be used by the AIS to heal ('redesign') itself. The GRT can be seen as a 'white-hat' hacker agent: it finds vulnerabilities in AISs.

In order to effectively evolve a population of red patterns, the GRT needs the AIS to provide a fitness value for a candidate vulnerability. The fitness value of a red pattern can be measured as its closeness to a state in which it can avoid detection by the AIS. In this dissertation, it is simply calculated as the number of detectors in the system that failed to detect it. If a red pattern is assigned a fitness of zero if it is part of self. So, a red pattern that avoids the detection of all the detectors (or has a fitness value that equals to the size of the detector set in the current AIS) potentially represents a vulnerability in the system.

The red patterns take the representation of vector of integers, as described in Chapter 3. The initial red pattern population is randomly generated. For each GRT iteration, an offspring is created by selecting two parents, *p1* and *p2*, from the population using binary tournament selection [Goldberg and Deb 1989] and mating the parents using uniform crossover[7] [Sywerda 1989]: each integer x_i in the offspring is randomly selected

[7] We chose the uniform crossover for the purpose of simplicity; other operator may improve the performance of the GRT.

from either x_{p1i} or x_{p2i}, where x_{p1i} and x_{p2i} are the corresponding integer values of the two parents. The resulting offspring is then mutated by having one of its integer replaced by a random number. The worst individual in the population is replaced by the offspring if the offspring has a better fitness value. However, if the newly generated offspring has a fitness that is equals to the size of current detector set, it will not replace the worst individual. Instead, it will be added to a vulnerability set. This way, we can use population sizes that are smaller than the number of requested vulnerabilities.

The GRT population is randomly regenerated if the GRT is not able to discover any vulnerability in a user-specified number of iterations. A counter records the number of unfruitful iterations during which no vulnerability is discovered, and this counter is reset when a vulnerability is discovered. This mechanism prevents the GRT from being trapped in an area that is too close to the self space or the coverage of existing detectors.

The GRT iterations go on until a user-specified number of vulnerabilities have been discovered or it has exceeded a user-specified limit on the number of population restarts. Meeting the second stopping criteria suggests that the current AIS is of certain security level so that the GRT has difficulty to find vulnerabilities in it. The vulnerability set will later be used by the GBT to redesign the AIS. Figure 4.1 shows the GRT algorithm in the form of a flow chart.

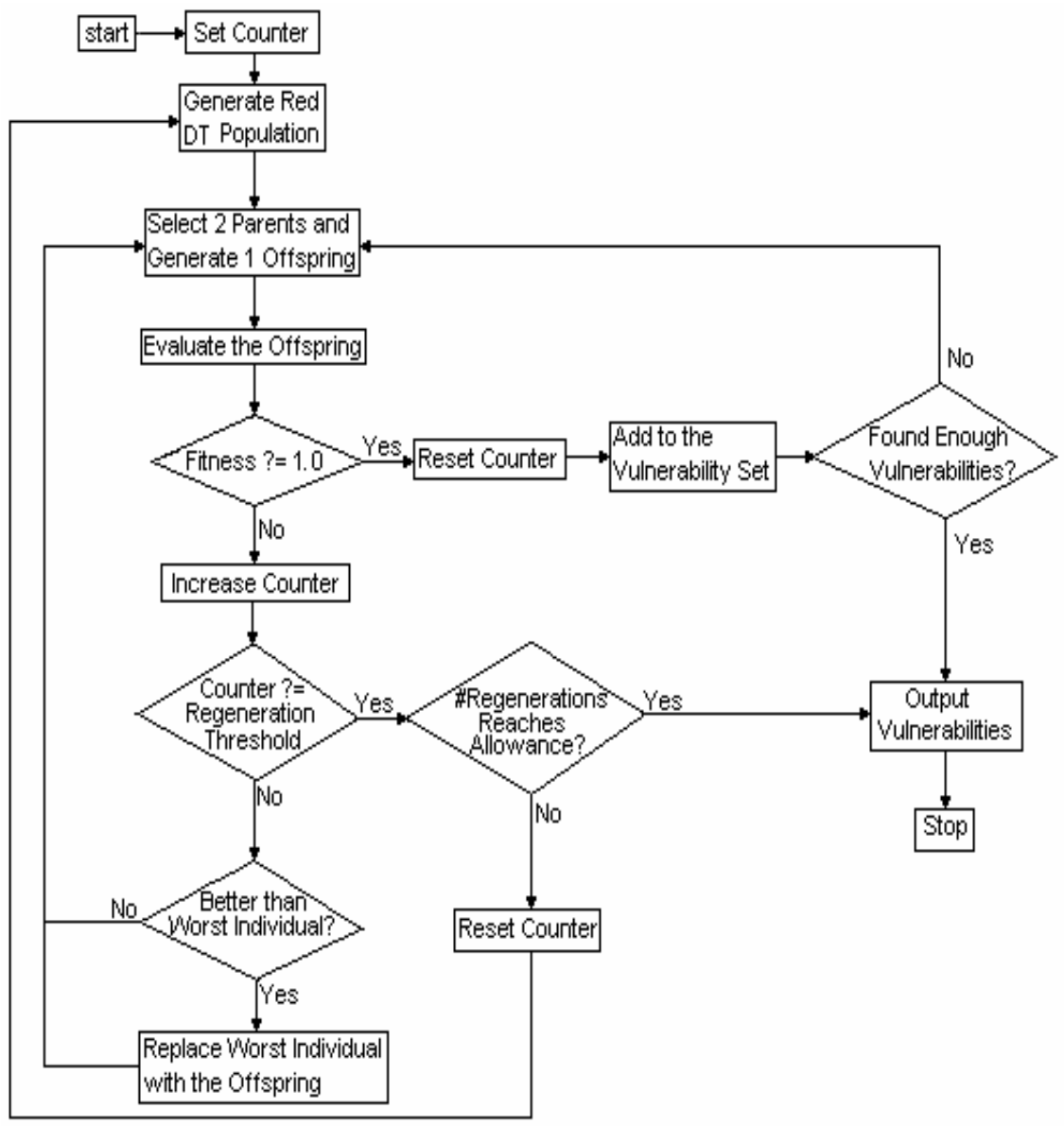


Figure 4.1: The flow chart shows the overall GRT algorithm.

4.3 GENERTIA Blue Team

As mentioned in the previous section, GENERTIA is a system used for AIS vulnerability analysis, design and redesign. It consists of two subsystems: the GRT and the GBT. The GRT is used to discover the holes in the AIS, and the GBT generates detectors that patch these holes, so the GRT and the GBT cooperate to strengthen an AIS.

The objective of the GBT is to initially design an AIS and then redesign it to make it more effective. The GBT takes a set of vulnerabilities discovered by the GRT and creates an additional detector set that covers these vulnerabilities. This additional detector set is then combined with the current detector set (which will result in a larger set of detectors).

The additional detector set is developed as follows. First, the GBT uses a steady-state GA to evolve a population of candidate detectors (CDs). These CDs are in the same representation as described in Chapter 3 and the initial population is randomly generated. Each CD is assigned a fitness based on the number of vulnerabilities it covers. If a CD matches a self pattern in the training set then it is assigned a fitness of zero.

Each time a CD is created with a fitness that is greater than the current best fitness, a release threshold counter is reset to zero. The release threshold counter is incremented on each iteration for which no better CD is found. When the release threshold counter

gets above a user-specified number of iterations (where there is no improvement in the fitness of the best fit CD), the CD with the best fitness is then removed from the population and added to the additional detector set. Once this has been done, the release threshold counter is reset to zero and a randomly generated CD is used to replace the removed CD.

This process of evolving, selecting the best fit CD, and adding it to the additional detector set is repeated until the additional detector set covers all of the vulnerabilities discovered by the GRT. At this point, the additional set of detectors is added to the detector population.

To prevent the GBT from running indefinitely, the GBT population is randomly regenerated if the GBT cannot find any valid detector during a predefined number of iterations. The GBT is only given a limited number of opportunities for population regeneration, and when this limit is reached, the GBT stops and all the remaining vulnerabilities are discarded. The difficulty of the GBT finding valid detectors for the remaining vulnerabilities suggests that these vulnerabilities are too close to the self space so that they probably should be regarded as generalization of self.

To reduce the detector set size, redundant detectors are not allowed. In order to accomplish this, when a CD is released, the vulnerabilities that are matched by this

detector are removed from the vulnerability set, so that the detectors released later will have different coverage. Figure 4.2 shows the GBT algorithm in the form of a flow chart.

The steady-state GA used by the GBT uses binary tournament selection [Goldberg and Deb 1989] to select two parents and applies uniform crossover [Sywerda 1989] to create one offspring[8]: each integer interval in the offspring is randomly selected from the two corresponding integer intervals of the two parents. The offspring then has one of its intervals (randomly chosen) mutated by randomly selecting new lower and upper bounds for the interval. In the experiments in this dissertation, the population size of the GBT was arbitrarily set to 50, and for each vulnerability set, the GBT was allowed 10 population regenerations including the initial generation.

[8] Again we used uniform crossover for simplicity; other operators may improve the performance of the GBT.

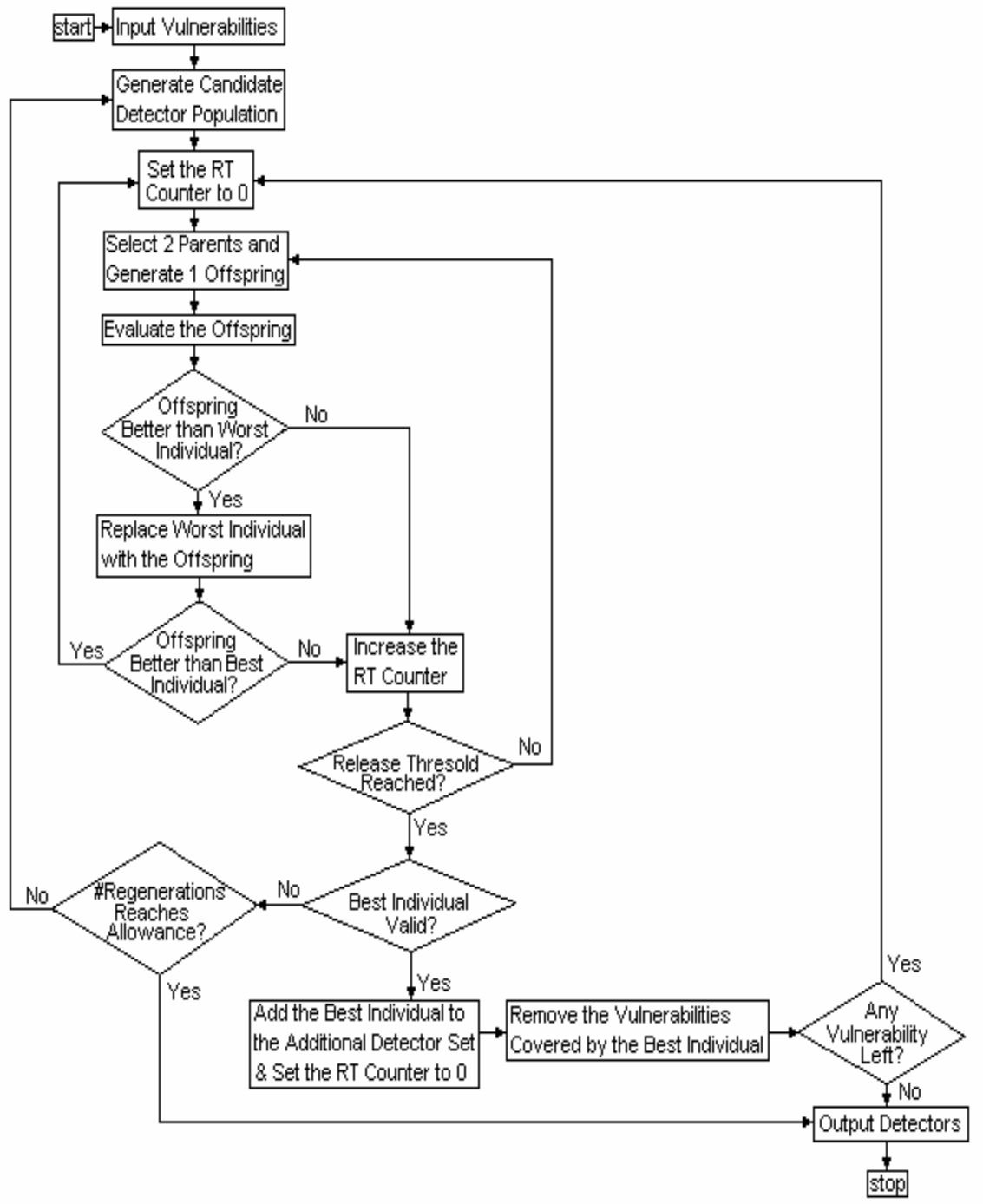


Figure 4.2: A flow chart shows the overall GBT algorithm. "RT" stands for release threshold

CHAPTER 5 A GENERTIA-BASED INTRUSION DETECTION SYSTEM

5.1 Introduction

Artificial Immune Systems have been employed to build systems for a broad range of applications [Dasgupta 1999], and because of the functional similarity between the immune system and network intrusion detection systems, significant researches [Hofmeyr et al.1998, Hofmeyr 1999, Hofmeyr and Forrest 2000, Kim and Bentley 2001b, Dasgupta and Gonzalez 2002, Harmer et al.2002] have been done to build AIS-based intrusion detection systems (IDSs). From an anomaly detection perspective, the functionality of an IDS is to classify the network traffic as either normal or abnormal. Abnormal traffic are potentially harmful to the network system or the hosts on the network. All the above mentioned AIS-based IDSs try to build a model (a detector set) of the network traffic for the protected network system by learning the normal patterns displayed by the system, and this model is later used to classify network packets as normal or abnormal.

Obviously the performance of an AIS-based IDS depends on the learning process in which the AIS is trained and the model is built. As mentioned in the previous chapter, the hypothesis developed through any learning algorithm has two types of errors: false

positives and false negatives. For intrusion detection, false negatives happen when abnormal packets are wrongfully classified as normal, and the consequence is the protected system being compromised.

No matter how well an IDS is designed and configured, vulnerabilities always exist and are subjected to exploit from hackers from all over the world. Vulnerabilities are usually discovered and patched after the protected system has been found compromised by hackers through these vulnerabilities in the IDS [Leon 2000]. So, to effectively reduce the probability of being compromised, an aggressively proactive IDS must seek to discover and patch the vulnerabilities before an attacker can exploit them. But how? In this chapter, we show that the GENERTIA system, proposed in Chapter 4, can be applied to strengthen an AIS-based IDS to reduce the number of vulnerabilities in the IDS.

Figure 5.1 shows the architecture of GENERTIA and its interaction with an AIS-based IDS for host-based intrusion detection. The GBT is used to design an IDS based on input from the network manager. After a preliminary host-based IDS has been designed, the GRT performs a strength and vulnerability analysis of the IDS, based on the input specifications of the network manager. The GRT analysis results in information concerning the relative strength of detectors (Labeled as RSD in Figure 5.1) comprising the IDS as well as a list of vulnerabilities (holes in the IDS). This information is then

given to the GBT to be used to redesign the IDS. The GRT and the GBT work cooperatively in a co-evolution fashion [Potter and De Jong 1998] to make the AIS more effective. The preliminary results presented in this paper are based on a host-based IDS. However, these results can be generalized to a network where each host has an instance of GENERTIA running on it.

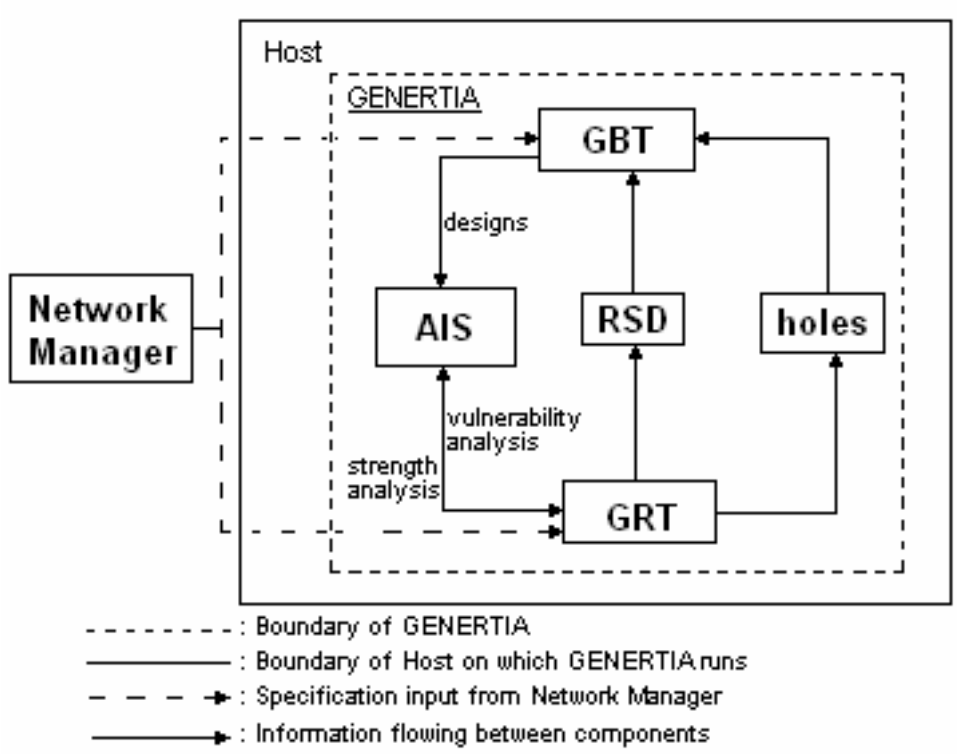


Figure 5.1 The architecture of GENERTIA for AIS-based IDS

5.2 Experiments

In this dissertation, the population size of the GRT was arbitrarily set to 50, and the vulnerability set size was arbitrarily set to 300, the GRT population regeneration

threshold was arbitrarily set to 5000, and the GRT was allowed 10 population regenerations including the initial generation.

5.2.1 Data Set

The purpose of this experiment is to show that the performance of an AIS can be improved by GENERTIA design/redesign.

The data set used in our experiments was taken from the MIT Lincoln laboratory 1998 DARPA Intrusion Detection Evaluation Data Set [Lincoln Lab 1998]. Although this data set has been criticized because it was not generated by actual users and has some weaknesses such as the lack of noisy traffic [McHugh 2000], we adopted this data set because it is publicly available and has been widely used for IDS performance testing. This data set contains 35 days of simulated network traffic, with each packet marked as background (normal) or attack (abnormal) packet. The packet volumes of the hosts that were subject to attack are listed in Table 5.1 (for the hosts on 172.16.115/116, the subnet traffic volumes are listed). From this data set, we filtered out packets involving port 80 (http[9]) and converted each packet into the patterns in the form as described in Chapter 3.

The patterns are implemented in the form (*ip_address*, *port*, *src*), where *ip_address* contains 4 integers that compose the IP address of the remote host, *port*

[9] We filtered out the traffic involving web server port number 80 because, according to [8], the traffic to this port continuously changes and does not have stable definition of normal behavior in term of data triple.

contains 1 integer and represents the port number on the receiving host, and *src* is assigned to 0 if the packet is incoming or 1 if the packet is outgoing. Such a pattern is referred to as a data triple (DT). The first 4 integers can take value between 0 and 255, and the fifth integer can take value between 0 and 69[10]. For example, a DT, (131.204.20.120, 21, 0), represents an incoming connection from a host with IP address of 131.204.20.120 to the local host through a port whose number is mapped to 21 (the ports were mapped into 70 categories according to [Hofmeyr 1999, page 46]). To obtain a host-based data set, the DTs involving host 172.16.112.50 were extracted. We chose this host because majority of the network traffic involves this host, and this host was the target of most attacks as shown in Table 5.1. We extracted the normal DTs and deleted all duplicates to obtain the self set. The self set consisted of 112 self DTs. This self set is used as a bound self space for the experiments (we assume that any other DTs are abnormal DTs).

Our AISs were trained on approximately 80% of the self set with 89 randomly selected self DTs. The remaining 23 self DTs were used to test for false positives. Our test set consisted of a total of 1604 distinct DTs, including all attack packets launched during the 35 day period. Note that there are 22 abnormal DTs also appeared in the

[10] In TCP/IP, 16 bits are used for port number, resulting in 64k different numbers; however, by separating the well-known or undefined ports into groups, these numbers may by mapped in to 70 categories [22, page 46].

normal set. This is because some attacks, such like port-scan and denial-of-service, may generate packets that also occur in normal traffic.

Table 5.1 Network Traffic Volume of the Hosts Subjected to Attacks in 1998 Lincoln Lab Dataset

IP Address	Normal Packets	Self DTs	Abnormal Packets	Nonself DTs
172.016.112.050	9954	112	1404320	1604
172.016.112.149	12656	68	70	15
172.016.112.194	13288	67	70	15
172.016.112.207	11407	68	70	15
172.016.113.050	4056	54	217447	369
172.016.113.084	12149	66	70	15
172.016.113.105	12038	66	70	15
172.016.113.204	12463	67	70	15
172.016.114.050	5937	93	27730	392
172.016.114.148	34111	94	71	15
172.016.114.168	12065	69	72	15
172.016.114.169	11846	69	71	15
172.016.114.207	12570	64	70	15
172.016.115.	610	56	12	3
172.016.116.	409	38	8	2

5.2.2 Vulnerability Analysis

In this experiment, we investigated the ability of the GRT to discover vulnerabilities. First, we trained our AISs to generate detector set of sizes taken from the set: {10, 20, 40, 80, 160, 320}. These detectors were randomly generated with negative selection. The

performance of the trained AISs was tested using the test set, and then we used the GRT to find 300 vulnerabilities in the systems. We compared the efficiency of the GRT with another 'red team' that created red DTs by random generation.

The experiment was carried out as follows: for a certain size of the detector set, an AIS was trained using a training set, the vulnerability analysis was then repeated 10 times (due to the initial random GRT population, the result varies for each analysis on the same AIS), and the average of the 10 runs was calculated for this AIS; this process was repeated for 10 AISs, and the average of the 10 systems was calculated. We then repeat the above process and calculation for the other 5 training sets. Table 5.2 shows the average result of this experiment. Column 1 is the size of the detector set. The detection rate (DR) and the false positive rate (FPR) are shown in Columns 2 and 3. The DR was calculated as the number of abnormal DTs detected divided by 1611(the number of abnormal DTs); the FPR was calculated as the number of normal DTs detected divided by 23 (the number of normal DTs that were not included in the training set). Columns 4 and 5 show the numbers of iterations needed to find the first vulnerability in the AIS, using the GRT and random generation respectively; the numbers of iterations needed to find 300 vulnerabilities in the AIS using the GRT and random generation are recorded in Columns 6 and 7 respectively. The numbers in the parentheses are the standard deviations.

The GRT used a population of 50 red DTs that initially are randomly generated, and we have counted the generation of each DT as one iteration, including the first 50 random DTs. So the 'real' GRT iteration begins with the 50th iteration, before when GRT has the same efficiency as the random red team. Accordingly, in the table, the numbers that are less than 50 indicate that vulnerabilities are discovered by random DTs.

Table 5.2 Comparing the efficiency of the GRT and random generation (The numbers in parentheses stand for the standard deviation)

Detector Set Size	DR	FPR	1 Vulnerability		300 Vulnerabilities	
			GRT	Random	GRT	Random
5	7.17%	0.94%	1(0.5)	2(0.5)	407(14.4)	458(19.7)
10	14.89%	2.46%	2(0.0)	2(0.0)	526(34.2)	658(58.7)
20	27.80%	4.06%	4(0.5)	3(0.5)	764(62.2)	1128(115.1)
40	43.94%	7.61%	8(1.6)	9(0.7)	1111(83.0)	2607(453.8)
80	60.69%	15.51%	26(3.4)	24(6.4)	1633(143.8)	6960(1805.7)
160	77.62%	22.83%	51(10.8)	80(18.3)	2162(203.6)	23953(5692.8)
320	91.52%	32.83%	102(14.3)	339(95.0)	3199(328.1)	113225(50185.3)

As one may expect, using the GRT is more efficient than random generation. One can also see that increasing the detector set size results in an increased number of iterations needed to find vulnerabilities. Obviously this is because increasing detectors results in a reduced number of vulnerabilities in the system.

5.2.3 Generating Detectors

In this experiment, the effectiveness of the GBT was investigated. One important parameter for the GBT is the release threshold. The release threshold defines the number of iterations the GBT needs to go through, during which a better CD cannot be found, before the best CD is added to the detector set. To investigate the effect of the release threshold on the efficiency of the GBT, different release threshold settings of 100, 200, 500 and 1000 were tested.

The experiment was carried out as follows: an AIS with 320[11] detectors were trained with a training set, then a GRT was used to discover 300 vulnerabilities in the system and a GBT was allowed to design detectors for the vulnerability set; this GRT/GBT iteration was repeated 10 times for this AIS and the average of the 10 runs is calculated; this process was repeated for 10 AISs and the average was calculated for the 10 AISs. Then the whole process and calculation was repeated with other 5 training sets; finally the average results from the 6 training sets was calculated.

Figure 5.2 shows the average GBT performance in terms of the number of detectors generated and the number of iterations taken to generate these detectors, with the x and y-error bars showing the standard deviation values. In this figure, each point

[11] In this and following experiments, we generated 320 detectors to keep the detector set size consistent with the settings we used in later experiments, in which we used $10*x^2$ (10, 20, 40, 80, 160, 320). It was also learnt from later experiments that for GENERTIA-redesigned AISs, with a release threshold of 1000, 320 is a rough limit on the size of the detector set, beyond which it will become difficult to generate more detectors.

represents a different setting of release threshold, the x-axis represents the size of the additional detector set generated to cover the discovered vulnerabilities, and the y-axis represents the GBT iterations needed to generate the additional detector set. One can see in Figure 5.2 that increasing the release threshold (RT) can lead to smaller additional detector sets. This is because the higher the release threshold, the better the chance of generating CDs that match more vulnerabilities, however, at the price of more GBT iterations as shown in the figure.

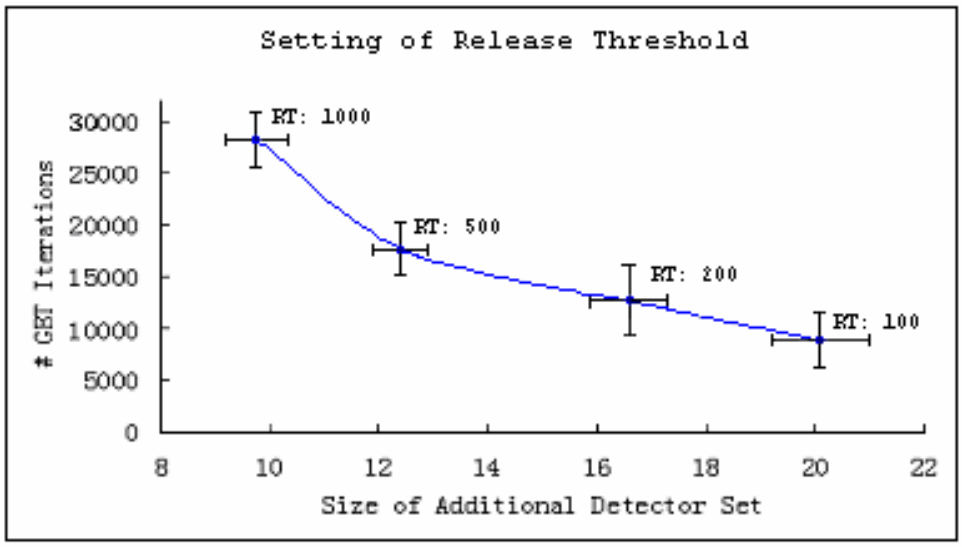


Figure 5.2 The effect of release threshold on the GBT activity. The Pareto front shows the trade-off between the size of the additional detector set and the GBT iterations needed to generate these detectors, with different GBT release threshold. The points on the curve represent the average values and the error bars show the standard deviation values.

The setting of the release threshold is not trivial. If the threshold is set too high, then the GBT tends to waste many iterations trying to search for a better CD even when the chance for finding such a detector may be small. This is especially true when the

vulnerability set contains DTs that are far away from each other. They can be separated by self DTs so that a single detector cannot cover red DTs without covering part of self. On the other hand, if the release threshold is set too low, the GBT tends to generate more CDs than are needed. Therefore, there is a trade-off.

In the following experiments, we set the GBT release threshold to 1000, which is relatively high, in order to generate fewer detectors, at the cost of more GBT iterations. Using a lower release threshold, GENERTIA should be able to more quickly discover/patch vulnerabilities, at the cost of a larger detector set.

5.2.4 GENERTIA Redesign

The objective of GENERTIA redesign is to make an AIS-based IDS more effective and more efficient as well. Effectiveness of an IDS is how accurately the system can detect intrusion, while the efficiency of an IDS is how economical the system can be in terms of computing resources (memory, CPU time etc.) consumed to perform intrusion detection. In this experiment, we evaluate the effectiveness of an AIS using the GRT effort for finding vulnerabilities in the system, and the associated efficiency using the size of the detector set in the system. The number of GRT iterations needed to discover the vulnerabilities in an IDS can reflect the effectiveness of the system because it shows how hard the GRT must work in order to discover vulnerabilities. The size of the detector set

can reflect the efficiency of the system because a smaller number of detectors require less memory and computation.

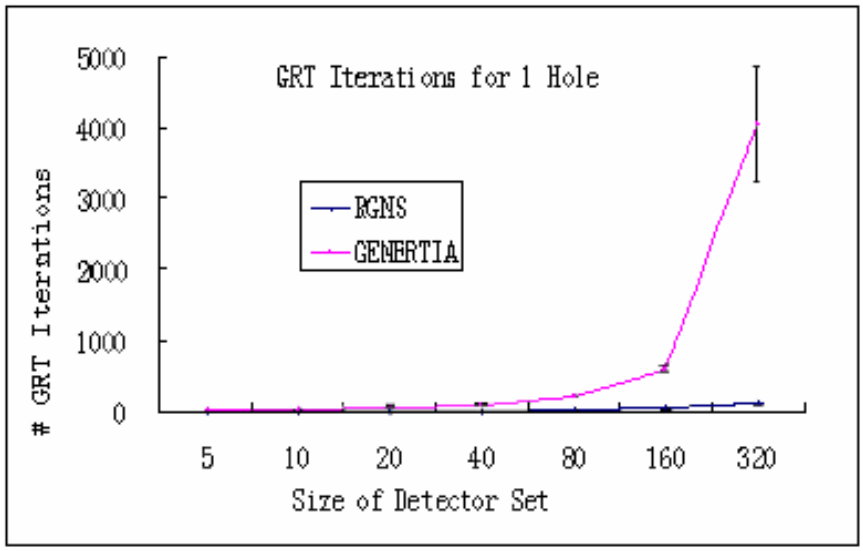
To show that a GENERTIA-redesigned AIS-based IDS can outperform ones without GENERTIA, experiments were carried out to compare the effectiveness of a system that used GENERTIA and one that did not. For the system that used GENERTIA, the AIS was trained to generate detector set of sizes taken values from the set: {10, 20, 40, 80, 160, 320}. First, the system generated only 1 detector. Then the GRT was used to discover 300 vulnerabilities in the system, and the GBT was allowed to generate detectors for these vulnerabilities. Next the generated detectors were added to the system's detector set. The GRT/GBT cycle was repeated continuously until the detector size reached the specified size. This experiment shows how GENERTIA can redesign a weak system (which has only 1 detector) to make it stronger. The GRT/GBT cycle is actually the redesign process although the whole process can also be viewed as the process of designing a system from scratch. We will refer to a GENERTIA-redesigned AIS as a redesigned system for simplicity.

For the AIS that did not use GENERTIA, the system was simply trained to generate RGNS detector set of sizes also taken values from: {10, 20, 40, 80, 160, 320}. We refer to this AIS as an RGNS system for simplicity.

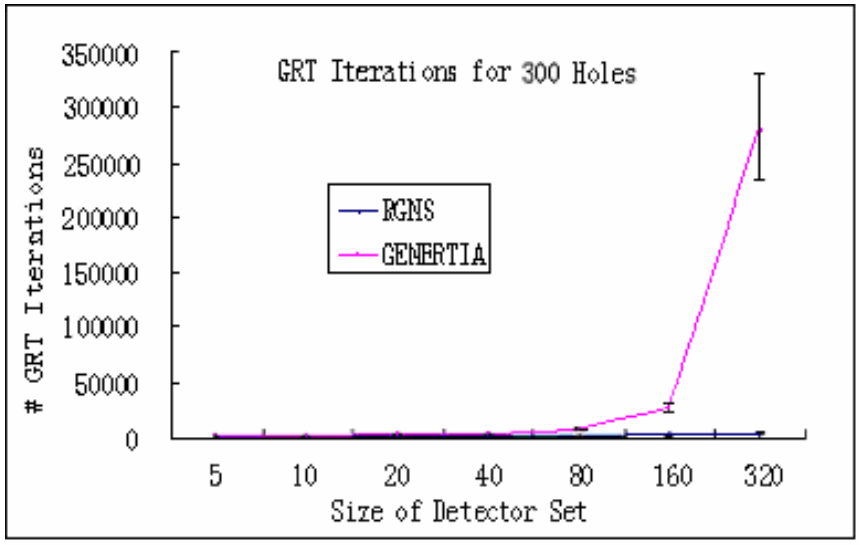
After the two systems were built the GRT was used to perform vulnerability analysis on the systems. With each of the 6 training sets, 10 AISs were built; and the performance of each system was determined by taking the average of 10 runs. So, the result reported here was the average of 600 runs for each parameter setting.

Figures 5.3a and 5.3b show the results of the experiments. The x-axis is the size of the detector set, and the y-axis is the number of the GRT iterations needed to discover the first (Figure 5.3a) and the 300th (Figure 5.3b) vulnerabilities in the system. From these figures, it is clear that for a system with larger detector sets, more effort was required to discover vulnerabilities. The reason for this is, as we mentioned in Experiment I, that larger detector sets have larger coverage of nonself space, which results in fewer vulnerabilities in the system.

However, by comparing the two curves in each figure, we can see that there was a big difference between the redesigned system and the RGNS systems. Although increasing the size of the detector set constantly improved the performance of both systems, in terms of the number of GRT iterations needed to discover vulnerabilities in the systems, the improvement process is much more dramatic in the redesigned system as GENERTIA-designed detectors were added to the system. For the redesigned system, the system effectiveness grew exponentially with the increase of detector set size. For the RGNS systems, the growth of system effectiveness was linear to the increase of the size of detector set. Comparing the redesigned system with the RGNS system, we found that when the size of detector set was 320, for the RGNS system, the average number of GRT iterations needed to discover the first and the 300th vulnerabilities were 102 (with standard deviation of 14) and 3198 (328), respectively; for the redesigned system, it required 4038 (819) GRT iterations to discover the 1st vulnerability and 283177 (48151) iterations to discover the 300th vulnerability.



(a)



(b)

Figure 5.3 Comparing the GRT effort needed to discover 1 (a) and 300 (b) vulnerabilities in systems trained with and without GENERTIA.

Detectors generated through GENERTIA design have different coverage. So for a GENERTIA-redesigned system, a larger detector set will have larger coverage. For the RGNS system, a larger detector set does not necessarily have larger coverage, because the randomly generated detectors may have large overlapping coverage with detectors

already existing in the system. We found that for a system with 10240 RGNS detectors, it usually took less than 2500 GRT iterations to discover a vulnerability in the system. Obviously, simply increasing the detector set size is not an efficient way to improve system performance. GENERTIA offers a more efficient approach to improving the performance of AIS-based IDSs by adding a minimal number of detectors for a maximal amount of coverage.

5.2.5 Performance Comparison of AISs with/without Redesign

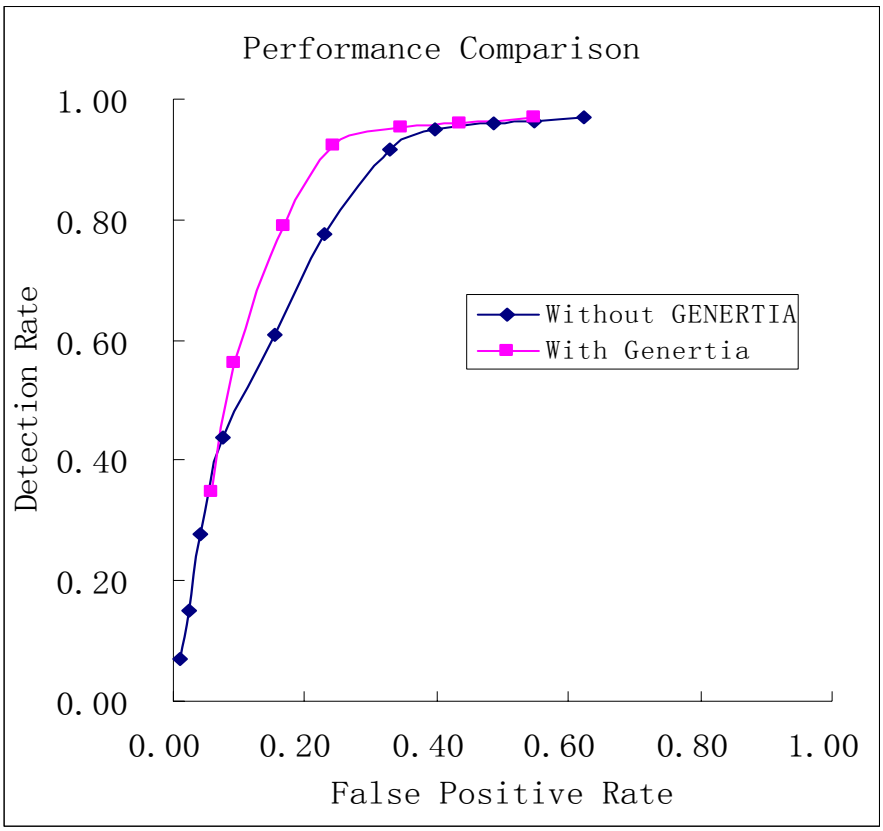
To better understand the effect of GENERTIA redesign on the performance on the AIS, experiments were carried out to compare a GENERTIA-redesigned AIS and an AIS without redesign.

Each experiment consisted of 2 stages: training and test. In the training stage, for the redesigned AIS, 1 RGNS detector was generated, then the GRT/GBT cycle was applied to design the rest of the detectors; for the AIS without redesign, it simply generated a defined number of RGNS detectors. In the test stage, the systems were subjected to the test (attack) set. We performed cross-validation by using the 6 training sets described in Section IV. For each of the 6 training sets, the experiment was repeated for 10 times, and the average performance of two systems during their 60 runs was used for comparison.

The performance of the systems was evaluated with respect to both detection and false positive rates. As mentioned in the first experiment, the detection rate was calculated as the number of abnormal DTs detected divided by 1611(the total number of abnormal DTs); the false positive rate was calculated as the number of normal DTs detected divided by 23 (the number of normal DTs that were not included in the training set). Figure 5.4 shows the average performance of the IDSs in the form of the receiver operating characteristics (ROC) diagrams, in which the performance of the systems is plotted as detection rate versus false positive rate. The y-axis represents detection rate, and x-axis represents false positive rate. The optimal performance (100% detection rate and 0% false positive rate) is located in the upper left corner.

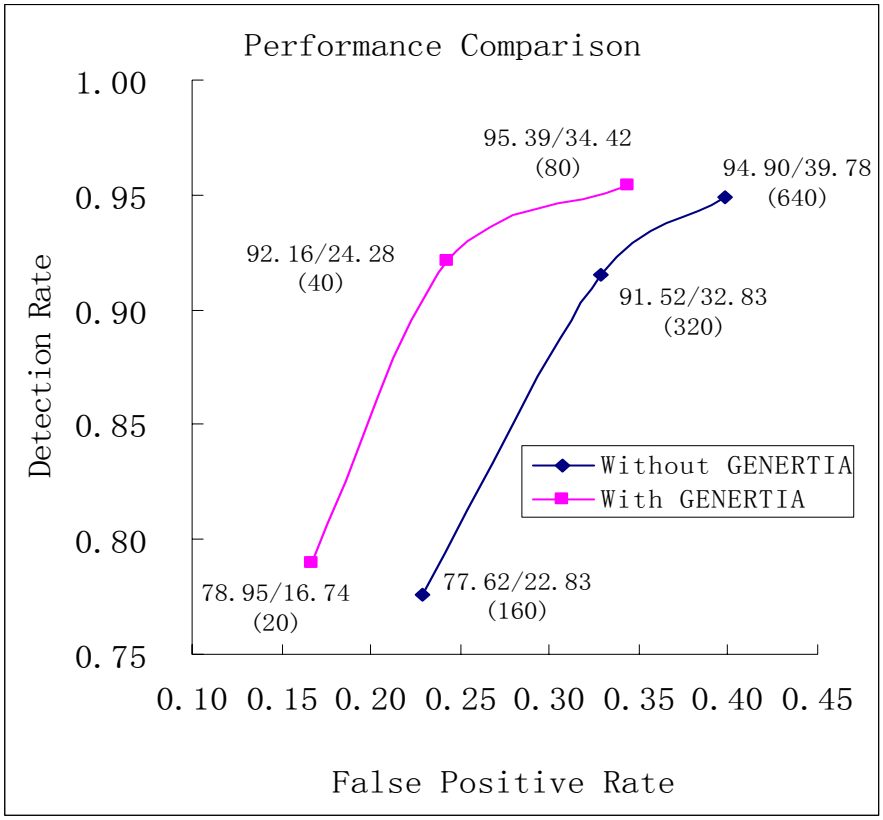
We speculate the reason why the GENERTIA-redesigned system outperformed the one without redesign as follows. GENERTIA is an interactive system. During the GENERTIA redesign process, vulnerabilities discovered by the GRT are only candidate 'holes', and they need to be confirmed to be real 'holes' in the system by the network administrator before detectors are designed to patch these holes. The reason why GENERTIA may increase detection rate and reduce false positive rate can be explained as follows. The DT space is divided into 3 parts: "self" represents the known self DTs (at the very beginning, it consists of all the DTs in the training set), "nonself" represents the DTs that are currently covered by the detector set and these DTs are deemed as 'known' nonself, "unknown" represents the DTs that are neither 'known' self nor 'known' nonself.

The vulnerabilities discovered the GRT are unknown DTs, and it is the responsibility of the network administrator to classify these DTs as self or nonself. If it is classified as self, it is added to the training set (becomes a known self); no detectors will be allowed to cover this DT, and the AIS will not generate false positive from this DT, so this tends to lower the false positive rate. If it is classified as nonself, then new detector is designed to cover this DT and this tends to raise the true positive rate (or detection rate).



(a)

Figure 5.4 ROC diagrams for AISs with/without GENERTIA redesign. (Part 1 of 2) (a) Full scale. In both curves, the points at the lower left corner represent a detector set of size 5; size of the detector set doubles at each point along the curve towards the upper right corner.



(b)

Figure 5.4 ROC diagrams for AISs with/without GENERTIA redesign. (Part 2 of 2) (b) Detail of the upper left corner, with the labels beside the curves showing the numbers in the form "Detection Rate/False Positive Rate (Detector Set Size)".

5.3 Discussion

5.3.1 The Importance of the Preliminary Work on GENERTIA

Implementing and maintaining secure computer systems is difficult, because there is no way of ensuring that a certain level of security has been achieved [Anderson et al. 1995, Hofmeyr 1999]. However, by iteratively applying vulnerability analysis and system redesign, we can attempt to minimize the chance of the system being compromised. The preliminary experimental results presented in this paper show that one can improve the

effectiveness of an immunity-based IDS by applying cycles of vulnerability analysis and the system redesign.

Lightweight IDSs (in terms of the size of detector/signature sets) are always desirable, regardless if it is signature-based or an anomaly detection system. An IDS with too many detectors, not only unnecessarily consumes more resources of the host on which it runs, but also slows down packet processing. When the IDS is overloaded by the network traffic, it creates a bottleneck, and this makes the IDS a vulnerable target of denial-of-service attacks. Reducing the size of the detector set is vital for an AIS to be useful. To become a memory detector, a mature detector usually requires co-stimulation from a network administrator [Hofmeyr and Forrest 2000, Harmer et al. 2002, Kim and Bentley 2004], so reducing the total number of detectors may save the administrator from a flood of co-stimulation requests. So a GENERTIA-redesigned AIS can alleviate the burden of the administrator by reducing the false positive rate.

GENERTIA provides a novel approach to the redesign/design of AIS-based IDSs. Although this work has focused on the network intrusion detection, however, the concept of GENERTIA may be applied to other anomaly detection systems that build and use models of normal behavior. In fact GENERTIA can also be applied to signature-based IDS to produce a hybrid system that perform both signature-based and anomaly detection. For example, GENERTIA can be applied to Snort for vulnerability analysis by changing

the GRT fitness function to the number of rules a red DT can avoid. The role of GRT could also be replaced by any attack/hacker software so that the system administrator could find out whether the system would be vulnerable to such attack. The concept of GENERTIA may also be used in the development of general machine learning applications.

5.3.2 Inefficiencies and Improvements of GENERTIA

At the price of a higher false positive rate, one can increase the detection rate of an AIS with a larger detector set as seen in Figure 5.3. However, with GENERTIA, the cost to generate additional detectors grows with the increasing in the number of existing detectors. This is because the existing detectors determine the number of and distribution of the vulnerabilities in nonself space. The vulnerabilities that are relatively far away from self are easily discovered by the GRT and patched by the detectors designed by the GBT in early stages. With the increase in the detector set size, the majority of the vulnerabilities in the system tend to be distributed around self space, which are difficult for the GRT to discover, and difficult for the GBT to design patching detectors.

To solve this problem, an easy approach is to 'fine tune' the match between the detector and the DT. Dasgupta and Gonzalez [Dasgupta and Gonzalez 2002] proposed a multi-level approximation approach to model nonself space that is in the vicinity of self space. Similarly, the GBT may design detectors with various matching threshold values,

with higher matching threshold to match the nonself DTs that are close to self space. In our prototype system, the GRT is implemented using a steady-state GA; however, it is possible to implement a GRT that alternatively applies different algorithms, such as a "genetic swarm" as suggested by Dozier [Dozier et al. 2004b]. Since different algorithms have different search behavior, combining a variety of algorithms may lead to a more thorough search [Dozier et al. 2004b]. The two GAs used by the GRT and the GBT used the uniform crossover operators. We chose these operators for simplicity rather for efficiency, and it is likely that some other crossover operators may improve the efficiency of the two teams. The parameters were set arbitrarily and optimization of these parameters may also improve this GENERTIA.

We have shown that GENERTIA can design/redesign lightweight IDSs, however, this is at the expense of computation time. We have not yet established the complexity expression for GENERTIA, however, we can give a rough analysis by comparing the time complexity of generating detectors by RGNS and GENERTIA. We may compare the two in term of the number of detector trials; however, the cost in generating a RGNS detector is very different from that of generating a detector through GENERTIA. So we compare the two in term of number of detector checks (NDC): the number of checks for whether a detector matches a DT. We have recorded NDC when performing the experiments described in the previous section, Table 5.3 shows the average NDC needed

to generate detectors with RGNS and GENERTIA. From this table, one can see that GENERTIA is significantly expensive than RGNS.

Table 5.3 Time cost in terms of NDC for generating detectors using RGNS and GENERTIA. (The numbers in the parentheses are standard deviation values)

Detector Set Size	RGNS	GENERTIA
5	1270(176)	936544(132023)
10	2602(195)	1571719(153587)
20	5221(208)	3436167(340054)
40	10451(533)	7163712(618000)
80	20937(909)	16113746(2732696)
160	42522(3082)	42315748(5016890)
320	84421(4914)	300251538(52273681)

With RGNS, the cost in computation time is linear to the number of detectors generated, because each newly generated detector has nothing to do with the ones already existing in the system; it goes through the same negative selection process independently, and statistically, costs the same in terms of computing power. With the experimental setup and parameter setting in our experiments, the cost for generating a RGNS detector is roughly 260 NDC. In the ideal case, the number of detector checks needed to generate a valid RGNS detector is the same as the size of the training set. For other failed trials, assuming the average NDC performed is the half of the size of the training set, we can easily calculate that the average number of detectors tried for successfully generating a RGNS detector is (260-89)/(89/2)+1 = 4.8.

With GENERTIA, the cost for generating detectors is from the cost for the GRT to discovering vulnerabilities and the cost for the GBT to design detectors for these vulnerabilities. The GRT uses a GA to evolve red DTs. During each GRT iteration, an offspring is generated and its fitness evaluated. NDC performed for fitness evaluation of each red DT equals to the size of the detector set. Since the number of GRT iterations increases exponentially with the increase in the size of the detector set, the NDC performed by the GRT increases in the same trend. The GBT also uses a GA to evolve CDs. During each GBT iteration, an offspring is generated and its fitness evaluated. To evaluate the fitness of a CD, detector checks are performed for the DTs in the training set and the red DT set from the GRT. The data for GENERTIA in Table 5.3 is plotted in Figure 5.5. From this figure we can see that the computation cost in terms of NDC increased exponentially with the increase in the size of the detector set. The trend of the curve is very similar to the ones in Figure 5.3, the GRT iterations needed to find vulnerabilities in an AIS. If we consider the GRT effort for finding vulnerabilities in an AIS the measurement of the strength of the system, then the resemblance in these curves suggests that the cost of building an AIS is proportional to the strength of the built system.

As a real time application, a successful IDS must have good scalability. Heavy computation needed by GENERTIA may seriously limit the feasibility of its application as an IDS redesign system. Improving the GENERTIA by design more efficient GRT

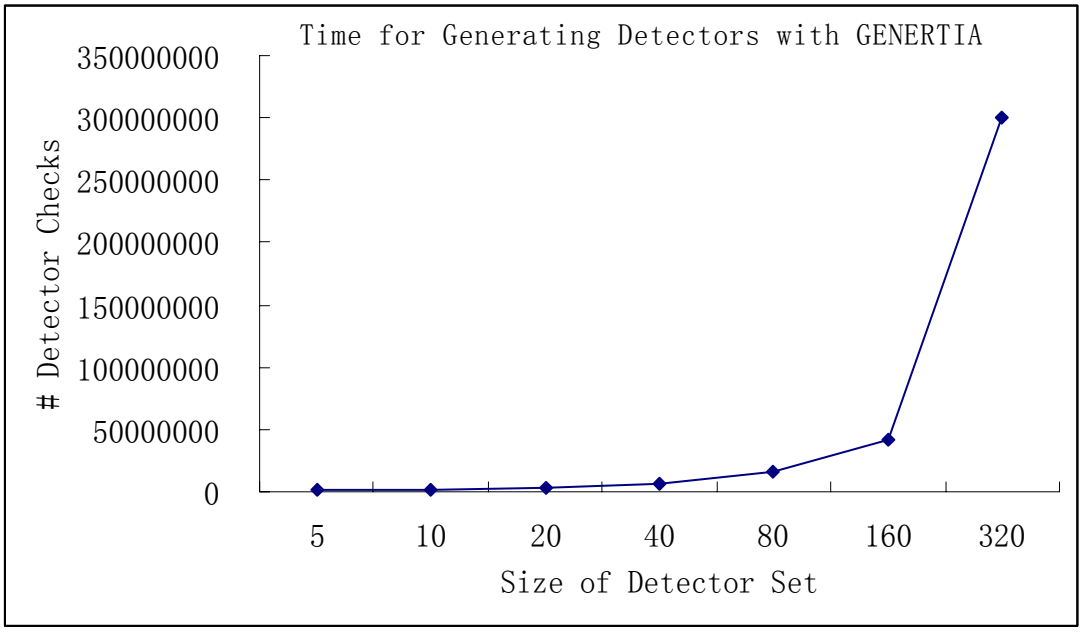


Figure 5.5 The average number of the GBT iterations needed to design detectors

and GBT will alleviate the problem. Besides, there is a way to reduce the computation time in running GENERTIA. We can simply set a limit on the computation time for the GRT. Rather than being given unlimited time to discover 300 vulnerabilities, the GRT can be given an iteration limit, and it terminates and hands off the discovered vulnerabilities when the limit is exceeded. This may reduce the number of discovered vulnerabilities and it is possible that no vulnerability is discovered at all using the given limit. On the other hand, however, no vulnerability being discovered in given limit may suggest that certain level of security is obtained. This provides a rough measurement to evaluate the strength of an AIS. As we have seen in Figure 5.5, the cost for building an AIS is proportional to the quality of the built system. If an optimum system is too

expensive to obtain, we have a choice for ordering a satisfactory system: a system that is resistant to the hacking by a GRT with a certain number of iterations.

5.3.3 Adaptation of GENERTIA to Dynamical Environment

The experiments conducted in this chapter are based on a static self set. However, the environment in which an IDS operates is continuously changing due to many factors such as security policy modification, computer configuration alternation and adding or deleting user accounts etc. An IDS must constantly and automatically adapt to these changes without severe performance degradation.

To design/redesign detectors for an AIS, GENERTIA need a set of sample normal patterns (training set). This training set may be initialized by collecting normal samples during an attack-free period or by collecting samples and filtering out abnormal samples. Each of the samples in the training set has a timestamp and when it is time-out, it is removed from the training set. Each time a same sample occurs on the network traffic the timestamp of the corresponding sample in the training set is updated (samples that occur on the network but do not appear in the training set may be deemed as generalization of the sample self set, although they may also originate from undetected attack packets). Each time a false positive occurs, the corresponding sample is added to the training set. This dynamics keeps the training set representative of the normal behavior of the network traffic during the past recent period.

GENERTIA, running as a separate process from the AIS-based IDS, does not need to be running constantly as the AIS itself. It may be scheduled to run at certain time interval, or it may be triggered when the size of the detector set (which contains only mature detectors) is dropped below certain threshold due to killing of detectors that cause false positives. This is comparable to the online adaptive training used by LYSIS [Hofmeyr and Forrest 2000], in which the immature detectors are exposed to the network traffic for a certain period for toleration, and the collection of the self samples occurred during this period correspond to the self sample set used for training in our AIS. The difference is that LYSIS uses online adaptive training, while our AIS uses offline adaptive training. One advantage of offline adaptive training is that the toleration of immature detectors is extremely shortened. Another advantage is that immature detectors are not tolerant to potential attack packets that avoid detection. With online adaptation, immature detectors have to be tolerant to all the packets that occur during the toleration period, including potential undetected attacks. So if an attack goes undetected, it may always goes undetected as long as it keeps appearance on the network traffic, because new detectors that detect it will be regenerated.

CHAPTER 6 AN IMMUNITY-BASED CLASSIFICATION SYSTEM

In this chapter, we propose the construction of a classification system based on anomaly detection that employs constraint-based detectors, which can be generated using a GENERTIA blue team.

6.1 Introduction

Classification is the mapping of data to a set of predefined classes. Usually, this problem is solved by developing a model that accurately classifies the given training data and then using the model to classify the target data. There are many approaches for developing a model [Dunham 2003]: statistic analysis [Hastie et al. 2001], neural networks [Orr 1996], decision trees [Quinlan 1986], rule-based algorithms [Witten and Frank 2000], distance-based algorithms [Kennedy et al. 1998] etc. All of them have some advantages and disadvantages.

The anomaly detection performed by AISs is essentially to solve a two-class classification problem: distinguishing nonself from self. The success of applying AISs to a broad range of anomaly detection application suggests a promising approach to design

immunity-based classification systems. The remarkable resemblance between an AIS and a classification system has been noted in [Hofmeyr and Forrest 2000]. However, there are several major concerns that have hampered the research in this direction:

1. Anomaly detection systems are practically two-class classification systems, while conventional classification systems usually solve multi-class classification problem.

2. Anomaly detection systems are usually trained using only negative (normal) samples in the training phase, while conventional classification systems are trained with samples from all classes.

3. Anomaly detection systems usually have many (hundreds of) anomaly detectors. The large number of detectors facilitates the effective distinguishing abnormal patterns from normal ones; however, from a classification perspective, the large number of detectors does not facilitate the abstracting of logical rules.

4. All learning systems have two types of errors: false positives and false negatives. Anomaly detection systems might have a bias of more tolerance for one type of error; for example, an anomaly-based intrusion detection system might be more concerned with false negatives (intrusions that are not detected) than with false positives (normal activities that are detected). Usually the two types of errors are equally undesirable in conventional classification systems.

It should be noticed that there are classification systems that are based on the Artificial Immune Network Model, for example, see [Watkins and Boggess 2002, Watkins et al. 2003, Hamaker and Boggess 2004]. However, these systems are practically based on K-Nearest-Neighbor approach.

This chapter presents the AIS-Based Classifier (ABC), a novel classification system based on an AIS that performs anomaly detection. Our experimental results on the benchmark Wisconsin Breast Cancer Dataset and Fisher's Iris Dataset show that the performance of our classification system is favorably comparable to some well-known classification systems on these datasets. Further experiments reveal that the GRT can be employed to quickly discover novel patterns that cannot be classified by the system. In other words, all the subsystems in the classifier cannot detect these patterns. This indicates that, although this classifier has high classification accuracy on these two datasets, there exist a large number of patterns that this classifier may not be able to correctly classify. These novel patterns may be used by the GBT to redesign the classifier to improve its performance. However, this is only possible if domain expert knowledge is available, because even though the GBT can be employed to design suitable detectors for the patterns, the newly designed detectors needs to be added to the right subsystems to detect these patterns.

6.2 An AIS-Based Classification System

6.2.1 Anomaly Detection-Based Classification

As mentioned above, anomaly detection can be viewed as a two-class classification problem: classifying a given sample as normal or abnormal. An anomaly detection system builds a model that reflects the characteristics of the normal samples. A multi-class classification system can be developed by combining multiple anomaly detection systems. For example, a three-class classification problem can be solved by building anomaly detection systems for the first two classes. Samples that belong to these two classes should be classified as normal by the respective systems, anything that is classified as abnormal by both systems should be considered belonging to the third class. Generally, an *n*-class classification system can be constructed with *n*-1 subsystems. It is also possible to build a subsystem for each class. This is especially useful if the test set may contain samples that do not belong to all the pre-defined classes and these samples will usually be misclassified as some class by traditional classification systems.

This approach seems circuitous to first map to an anomaly detection problem (to distinguish one class from all the rest) and then iterate on the multi-classes that remain, however, the whole process is straightforward: each subsystem is built the same way except that different training sets are used for training. The system is also expected to have high accuracy because each classification has to be agreed upon by all subsystems,

which provides a crosscheck mechanism. There are also some other advantages for adopting this approach. For example, the Fisher's Iris data contains sample measurements of three types of iris flowers, classification systems trained on these samples may correctly classify unseen samples from these three types of flowers, however, a sample from a fourth type of iris flowers would be wrongfully classified as one of the three types, while an anomaly detection-based classification system has chance to single out these samples as novel flowers.

6.2.2 Training of Subsystems

A typical anomaly detection AIS generates detectors using the Negative Selection Algorithm. The performance of systems trained this way depends on the size of the detector set: larger detector set may cover larger area of nonself space thus detects more anomalies. There are several problems with this approach: 1. this algorithm only uses normal samples while both normal and abnormal samples are provided for classification tasks; 2. the proper size of the detector set is not easily estimated; 3. the larger the detector set the more false positives may be generated, which results in more classification errors; 4. large number of detectors may be undesired compared to a compact set if logical rules are to be extracted from the detector set.

The proposed novel classification system, Anomaly Detection-Based Classifier (ADBC), employs a genetic algorithm for detector generation addresses the problems described above. With this approach, each subsystem is built by evolving a population of candidate detectors using a GENERTIA blue team as described in Chapter 4, and it is reviewed in the following paragraphs.

First, the subsystem randomly generates a population of candidate detectors. Then the population is evolved to search for valid detectors using a training set. The training set contains both normal and abnormal samples. A valid detector matches some abnormal samples in the training set, but does not match any normal samples in the training set. The fitness value of a candidate detector is the number of abnormal samples it matches, or zero in the case it matches any normal sample.

During each iteration, using binary-tournament selection, two candidate detectors are allowed to produce offspring using uniform crossover, then the resulted offspring has a randomly selected interval replaced with a randomly generated interval. The offspring replaces the worst individual in the population if the offspring had a better fitness value. For a predefined number of iterations (termed release threshold), if one candidate detector remains the best individual in the population and it had a fitness value greater than zero, then this candidate detector was released from the population as a valid detector. During a release, the valid detector is replaced with a randomly generated

detector, and the abnormal samples that were matched by the valid detector are removed from the training set. The whole process continues until there are no abnormal samples left in the training set. All the released detectors compose the detector set of the subsystem. This detector generation approach is depicted in Finger 6.1.

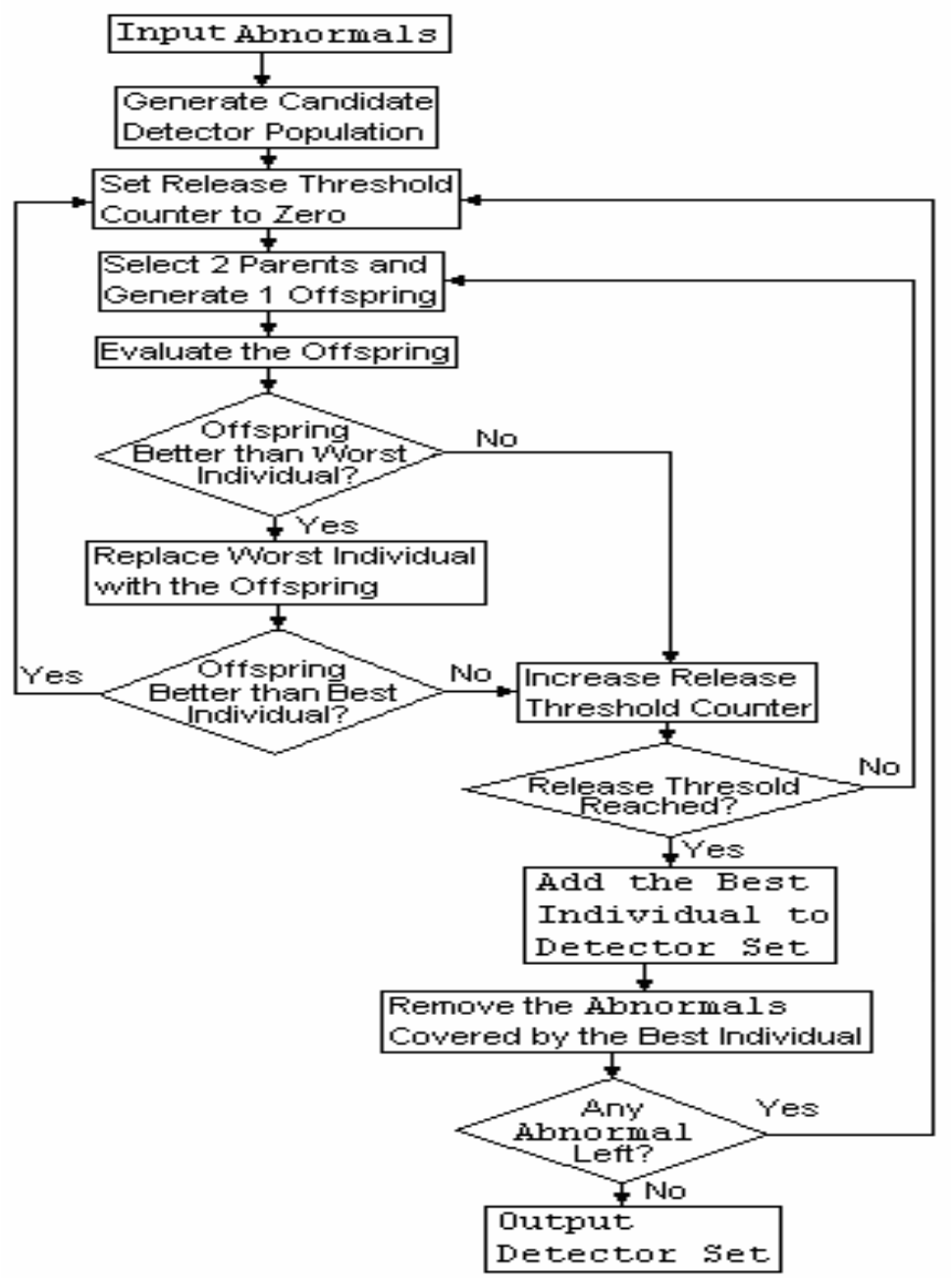


Figure 6.1 Flow chart of the algorithm used to generate detectors for classification system

6.2.3 Classification

After all the subsystems are trained, they are combined to form a multi-class classification system. In the classification process, each sample is subjected to these subsystems and the subsystems classified it as normal or abnormal independently. The classification is decided by comparing the results of the subsystems. For each sample, if one subsystem classifies it as normal, and the other subsystems classify it as abnormal, then this sample is classified to the corresponding class by the subsystem that claims it normal. In any other cases, the sample is not classified. These cases happen in following situations: 1. More than one subsystem classify a sample as normal. This situation usually suggests an overlap of the normal sample spaces of the corresponding subsystems. 2. All the subsystems classify a sample as abnormal. This happens when one subsystem misclassifies a normal sample as abnormal. In rare situation, this might suggest the sample belong to a novel class rather than the pre-defined classes.

6.3 Experiments

In this section, we test the proposed classification system using two well-known benchmark classification problems: Wisconsin Breast Cancer Dataset and Fisher's Iris Dataset. Both sets are taken from the Machine Learning Repository of the University of

California at Irvine[12]. In the following experiments, the release threshold of the genetic algorithm is arbitrarily set to 5000.

6.3.1 Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer Dataset contains sample data collected from University of Wisconsin Hospitals [Wolberg and Mangasarian 1990]. For each sample, 9 features are assessed and a value between 1 and 10 is assigned to each feature, with value 1 corresponding to a normal state and 10 to a most abnormal state. Therefore, each sample is recorded as a 9–dimensional vector of integers. There are 699 samples in this data set including 458 benign and 241 malignant samples, and these two types of samples are not linearly separable. Among these samples, 16 samples are incomplete with missing feature values. Although some classification systems simply discard the samples with missing values, we have arbitrarily replaced all the missing values with random values.

This is a two-class classification problem, which can be easily converted into an anomaly detection problem: given a set of benign samples, generate a system to detect samples having characteristics different from the characteristics of the given benign samples. Such an anomaly detection system can be built using the Negative Selection Algorithm (NSA). Since both benign and malignant samples are available for training, we can also build a system by generating detectors using the genetic algorithm described in Section 3. In this experiment, we test the performance of the constraint-based system

[12] http://www.ics.uci.edu/~mlearn/MLRepository. html

using both the NSA and the genetic algorithm. Since the existence of polarity (see Chapter 4), detectors with linear intervals are used for this data set.

For system training using the NSA, there are two parameters to set: the size of the detector set, and the matching threshold, *r*. We test the system with varied sizes of detector set. The *r* value is set to 5 which is found to be a good choice after we try other possible settings. The experiment was repeated for 100 times. During each run, a randomly selected 90% of the dataset was used for training and the rest 10% was used for testing. The experiment results in Table 6.1 are the averages of 100 runs for each parameter setting.

Table 6.1 Performances of AISs Trained with the NSA on the Wisconsin Breast Cancer Dataset

Size	False Benign (Negative)	False Malignant (Positive)	Total Errors
100	14.8 (4.3)	1.7 (0.8)	16.5
200	4.4 (3.4)	2.6 (1.1)	7.0
400	1.9 (1.6)	4.6 (1.5)	6.5
500	1.2 (0.8)	4.7 (2.3)	5.9
1000	0.2 (0.4)	5.8 (2.7)	6.0
2000	0 (0)	6.9 (2.9)	6.9

In Table 6.1, the numbers in the column labeled "Size" indicate the size of detector set of each subsystem rather than the total number of detectors in the system. False benign is a false classification of a benign sample as malignant, while false

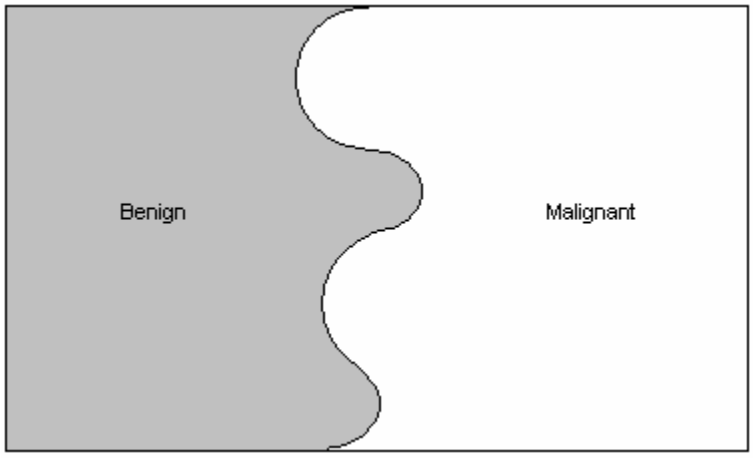
malignant is a false classification of a malignant sample as benign. Total number of errors is the sum of the two. The numbers in the parentheses are the standard deviations. From this table we can see that, the performance of systems trained with the NSA is affected by the size of the detector set. Increasing the size reduces false benign but also increases false malignant. The best performance of the system is obtained with a detector set of size around 500 for each subsystem.

The system built with detectors generated using the genetic algorithm has better performance. In our experiment with 100 runs, the average size of the detector set is 6.2 (standard deviation of 0.6), the average number of false benign is 2.1 (1.2), and average false malignant is 1.7 (1.3). These results indicate that the anomaly detection system built using the proposed algorithm is more accurate than the system simply trained with the NSA. A major reason is that our system is trained with both negative and positive samples while the system trained with the NSA only uses negative samples in the training. However, we have results that are not reported here indicate that a system trained with the genetic algorithm using fewer samples may still outperform the system trained with the NSA. So there are some other reasons for the performance difference.

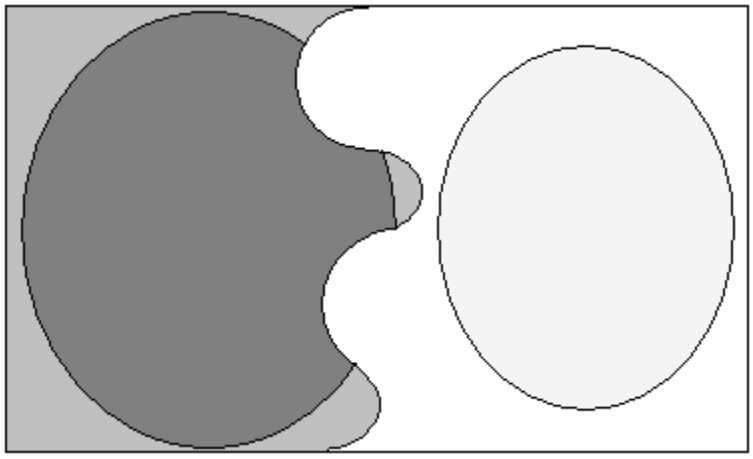
We speculate another reason that is depicted in Figure 6.2. A two-dimensional universe is used for simplicity. The universe can be divided into two spaces: benign and malignant, which are not linearly separable as shown in Figure 6.2a. Part of each

subspace is available for training an anomaly detection system, so we further divide the universe into four subspaces: known benign, unknown benign, known malignant and unknown malignant as shown in Figure 6.2b. Our goal is to develop a set of detectors with known benign and known malignant, and ideally, the detectors cover malignant space but do not cover benign space. The training process forces detectors away from the known benign and it is expected that this trend will generalize to unknown benign. However, the resulting detectors will always cover part of unknown benign because the known benign usually are not totally representative of the benign space. With the NSA, only the known benign samples are used for training, and the generation of each detector is independent of other detectors. The size of each detector is randomly decided and usually a large number of detectors are needed for a high coverage of malignant space. The generated detectors can be highly overlapped and spread over the entire universe except the known benign as shown in Figure 6.2c, which corresponds to a high malignant detection rate and a high false malignant rate as well. With the genetic algorithm, both known benign and malignant are used for evolving detectors. The genetic algorithm evolves detectors that has large coverage of known malignant and stops when all the known malignant are covered by the valid detectors, which results in a compact detector set with large coverage of malignant space as shown in Figure 6.2d. The known malignant also serves as a guide for the placement of the detectors so that detectors will

not be generated to cover certain benign space that is not represented by the known benign such as the left corners. This effect may greatly reduce the false malignant rate.

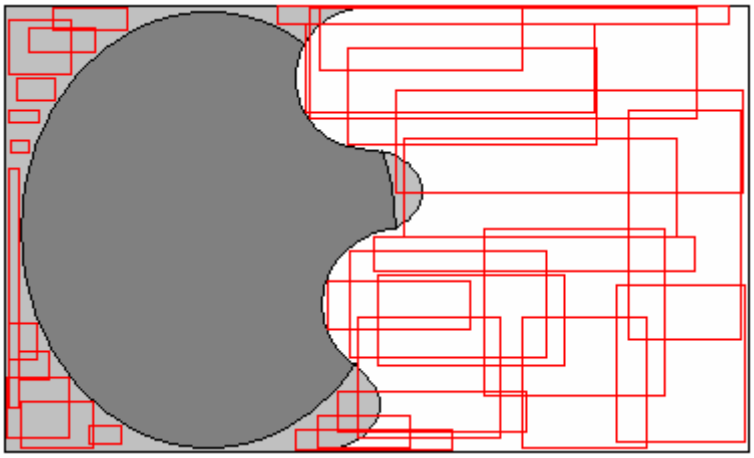


(a) The universe is divided into two spaces: Benign and Malignant;

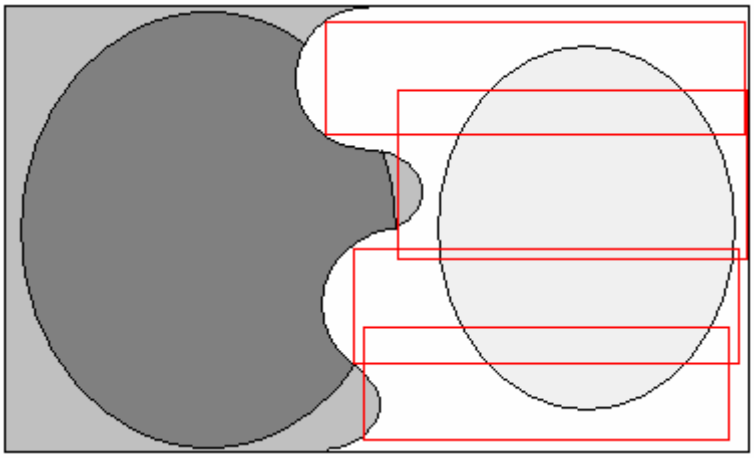


(b) Parts of both the benign and the malignant spaces are available for training;

Figure 6.2 The difference in placement and coverage between the detector set generated by the NSA and the detector set evolved using the genetic algorithm. (part 1 of 2)



(c) The detectors generated by the NSA have varied size of coverage, and a high coverage of the malignant space depends on the large number of detectors;



(d) The detectors evolved by the genetic algorithm tend to have large size of coverage and a high coverage of the malignant space is obtained if the known malignant is representative of the whole malignant space.

Figure 6.2 The difference in placement and coverage between the detector set generated by the NSA and the detector set evolved using the genetic algorithm. (part 2 of 2)

With the compact detector set, the anomaly detection system can be easily converted into rule-based classification systems. In this experiment, we can obtain a classification

system with 6 rules, with condition part of each rule taking the form of disjunctive feature constraints:

$x_0 \in lb_0 \,..ub_0 \vee ... \vee x_i \in lb_i \,..ub_i \vee \ldots \vee x_n \in lb_n \,..ub_n$

where x_i is the ith feature of a sample, and lb_i and ub_i are the lower and upper bounds of an interval for the ith feature. A sample is classified as malignant if at least r features of this sample satisfy the n feature constraints; if this sample is not classified as malignant by any rule of the rule set, it is classified as benign. As an example, the rule, <7..10 7..10 2..10 10..10 8..10 2..10 4..10 1..10 2..9>, classifies 201 malignant samples in the dataset. These rules are *M*-of-*N* rules using the classification rule terminology. These rules may facilitate data analysis for finding a logical description of the data. The system trained using the NSA can also be converted to a rule-based classification system; however, since the performance of this system depends on a large size of detector set, the conversion will result in a large number of rules. The accuracy of a classifier is calculated as the percentage of samples correctly classified by the classifier. The average accuracy of our classifier is 99.46%. For better understanding of the performance of this anomaly-based classifier, we compare it with some well-known classifiers in Table 6.4 (on page 106). These results are directly from [Duch 2002a-b]. We have inserted our system into the table according to the accuracy ranking, and we can see that the performance of our classification system is quite favorably comparable to the well-known classifiers.

6.3.2 Fisher's Iris Dataset

The Fisher's Iris Dataset contains three classes of iris flowers: *Setosa*, *Virginica* and *Versicolor*. There are 50 samples for each class. Each sample in the data set is defined by 4 measurements: petal length, petal width, sepal length and sepal width. *Setosa* is linearly separated from other two classes, while *Virginica* and *Versicolor* have considerable overlap.

The goal of the genetic algorithm (GA) is to generate a small set of detectors that has high detection rate and low false positive rate as well. To our knowledge, currently, there is no literature available that describes an anomaly detection based classification system, but the Fisher's Iris Dataset has been constantly used for testing performances of various anomaly detection algorithms. Since each subsystem of our classification system is actually an anomaly detection system, we can compare the performance of the subsystems with the available literatures.

The Variable-sized detector generation algorithm (here we refer it as the VGA) [Ji and Dasgupta 2004b], is proposed as an augmentation of the NSA anomaly detection. With V-detectors, the universe is defined as a normalized *n*-dimensional unitary space, $[0, 1]^n$, and detectors are defined by an *n*-dimensional vector that corresponds to the center and by a real value that represents its radius. Sample-detector matching is defined by the Euclidean distance between the sample location and the center of the detector and by the

radius of the detector. The VGA randomly generates a detector center that lies outside self space then continuously increases the radius of the detector until it contacts with self region. The algorithm terminates when a predefined number of detectors are generated or predefined nonself coverage has been reached by some simple statistic analysis. The most interesting characteristic of the V-detectors is their variable sizes of coverage, which is also with the constraint-based detectors, except that V-detectors are virtually hyperspheres while constraint-based detectors are hypercubes. The VGA uses only normal samples for training which corresponds to an anomaly detection problem definition where usually only normal samples are available.

We tested the performances of the three subsystems. For each subsystem, the samples in one class of iris were used as normal samples, and the samples in the other two class of iris were used as abnormal samples, as described in [Ji and Dasgupta 2004b]. The comparison between the two algorithms provides a better understanding of the performance of our algorithm. In [Ji and Dasgupta 2004b], 50% and 100% of the normal samples were used for training in two experiments. In our experiments, 33%, 50% and 100% of the whole dataset were used for training. The performances of the V-detectors and the constraint-based detectors generated using the GA are compared in Table 6.2. The results for V-detector are directly from [Ji and Dasgupta 2004b]. In this table, column of "Detector Type" indicates the type of detectors (variable-sized detector or constraint-based detector) and the percentage of samples used for training; for V-detector,

the percentage means the portion of normal samples used for training, while for C-detector, the percentage means the portion of all samples used for training (consisting of equal amount of normal and abnormal samples). The column of "Iris Class" indicates the individual subsystems. For example, for the *Setosa* subsystem, the *Setosa* samples were used as normal samples, and samples in other two classes were used as abnormal samples. The column "DR(%)" indicates the detection rate, which was calculated as the number of abnormal samples detected by the corresponding subsystem. The column "FPR(%)" indicates the false positive rate, which is calculated as the number of normal samples detected by the corresponding subsystem. The column "Detector Set Size" indicates the number of detectors that were generated by the VGA or the GA.

From this table, one can see that the genetic algorithm is capable of generating a very compact detector set with detection and false positive rates favorably comparable to the V-detector set. One reason is that we have used more samples for training (both normal and abnormal samples were used while VGA only uses normal samples). However, in the case where we used 33% of the whole dataset, which corresponds to the number of samples of 100% of a normal set, the overall performance of the three subsystems is still better than the three V-detector systems (considering both the detection rate and the false positive rate), especially in terms of detector set size. We believe the speculation depicted in Figure 6.2 for the NSA is also applicable to the VGA, except that V-detectors

are virtually hyper-spheres rather than hypercubes and that the VGA has a mechanism to increase the size of the coverage of V-detectors.

In the next experiment, we test our immunity-based classification system, namely, we will combine the anomaly detection subsystems as a three-class classification system for the Fisher's Iris Dataset. The classification system consists of three subsystems, one for each class of iris flower. Again, for comparison reason, we build classification systems

Table 6.2 Comparison of Performance of Varied-Sized and Constraint-Based Detectors Using Fisher's Iris Dataset

Detector Type	Iris Class	DR (%)	FPR (%)	Detector Set Size
V-Detector (50%)	Setosa	99.97	1.32	20 (7.87)
	Versicolor	88.3	8.42	153.24 (38.8)
	Virginica	93.58	13.18	218.36 (66.11)
V-Detector (100%)	Setosa	99.98	0	16.44 (5.63)
	Versicolor	85.95	0	110.08 (22.61)
	Virginica	81.87	0	108.12 (30.74)
C-Detector (33%)	Setosa	98.8	0.4	1 (0)
	Versicolor	95.5	5.6	1.8 (0.4)
	Virginica	95.9	6.2	1.5 (0.7)
C-Detector (50%)	Setosa	98.9	0.2	1 (0)
	Versicolor	96.0	2.4	2.4 (0.5)
	Virginica	98.4	6	1.6 (0.7)
C-Detector (100%)	Setosa	100	0	1 (0)
	Versicolor	100	0	4.6 (0.5)
	Virginica	100	0	3.6 (0.8)

using both the NSA and the genetic algorithm. The matching thresholds for the detectors of the three subsystems are set to 3, which appears to be a proper choice after testing other possible settings. For each subsystem, samples in one class are treated as normal set, and the samples in the other two classes are treated as abnormal set. For each experiment, a randomly selected 90% of normal and abnormal sets are used for training, so for each subsystem, 90 abnormal samples and 45 normal samples are used for training.

The first system is built by generating detectors for each subsystem using the NSA. The average performances over 100 runs of each parameter setting are shown in Table 6.3. The numbers in the column labeled "Size" indicate the size of detector set of each subsystem rather than the total number of detectors in the system. The numbers in the parentheses are standard deviation values. There are two types of errors: unclassified and misclassified, and the total of the two decides the accuracy of the system. Samples are unclassified when there are conflicts between classification results of any two subsystems, namely, more than 2 subsystems claim the sample to be normal, or all three subsystems claim the sample to be abnormal. Samples are misclassified when it is classified to be a class other than the actual class it belongs to. Misclassification is rare, because it indicates that two subsystems simultaneously misclassify this sample: one subsystem declares it abnormal but it actually does belong to this class, while another subsystem declares it normal but it actually does not belong to that class.

From this table, we can see that the accuracy of the system is primarily decided by the number of unclassified samples. The size of the detector set has tremendous effects on the accuracy of the classification. Increasing the size of the detector set reduces the number of misclassification but it may increase the number of unclassified samples. The best performance of the system is obtained when the size of detector set is around 400 for each subsystem. Before the size reaching 400, increasing detectors reduces unclassified samples because this reduces the number of abnormal samples that are not correctly detected with fewer detectors. After the size reaching 400, increasing detectors increase unclassified samples because more normal samples will be detected. This leads to the case where all subsystems declare these samples as abnormal and the system cannot classify them.

The second classification system is built with detectors generated using the genetic algorithm. In our experiment with 100 runs, the average sizes of the detector sets are 1 (with a standard deviation of 0), 4.2 (1.1) and 3.4 (1.0) for the subsystems of *Setosa*, *Versicolor*, and *Virginica*, respectively, the average number of unclassified samples is 1.1 (0.9), and the average number of misclassified samples is 0.4 (0.5). Therefore the average number of errors of the system is 1.5 (1.0), which corresponds to an accuracy rate of 99%.

Table 6.3 Performance of AISs Trained with the NSA on the Fisher's Iris Dataset

Size	Unclassified	Misclassified	Total Errors
100	13.6 (3.3)	0.4 (0.9)	14
200	9.8 (2.9)	0.1 (0.3)	9.9
400	8.6 (3.0)	0.2 (0.4)	8.8
500	9.3 (2.8)	0 (0)	9.3
1000	9.6 (2.5)	0 (0)	9.6
2000	10.6 (1.8)	0 (0)	10.6

For better understanding of the performance of this immunity-based classification system, we compare it with some well-known classifiers in Table 6.4. These results are directly from [Duch 2002a-b]. We have inserted our system (termed ADBC, standing for Anomaly Detection Based Classifier) into the table according to the accuracy ranking. From this table one can see that the performance of our classification system is quite favorably comparable to the well-known classifiers.

Table 6.4 Comparison of Classifier on the Wisconsin Breast Cancer Dataset and Fisher's Iris Dataset

	Wisconsin Breast Cancer		Fisher's Iris	
1	ADBC	99.46%	Grobian (rough)	100%
2	C-MLP2LN	99.0%	ADBC	99.0%
3	FSM	98.3%	SSV	98.0%
4	C4.5	96.0%	C-MLP2LN	98.0%
5	RIAC	95.0%	PVM (2 rules)	98.0%
6			PVM (1 rule)	97.3%
7			AIRS	96.7%
8			FuNe-I	96.7%
9			NEFCLASS	96.7%
10			CART	96.0%

Our classification system for the Iris dataset can also be converted into a rule-based system that consists of three set of rules. First, detectors of each of the three subsystems are converted to a set of rules. For example, detectors from the *Setosa* subsystem are converted into *M*-of-*N* rules that any sample satisfies one of these rules are classified as "not-*Setosa*". The resulting rule-based system will consists of three set of such rules for the three classes respectively. As an example, the following is three sets of rules we obtained that will classify the Iris dataset with 100% accuracy:

Set 1:

Rule 1: <52..79 20..42 24..64 6..23> → ¬*Setosa*

Set 2:

Rule 2: <43..79 20..44 10..20 19..25> → ¬*Versicolor*

Rule 3: <43..51 24..36 54..69 14..22> → ¬*Versicolor*

Rule 4: <51..66 28..30 49..52 18..22> → ¬*Versicolor*

Rule 5: <59..64 22..23 49..52 18..23> → ¬*Versicolor*

Set 3:

Rule 6: <43..77 32..42 11..49 1..16> → ¬*Virginica*

Rule 7: <52..70 24..40 33..39 16..17> → ¬*Virginica*

The conditional part of each rule is interpreted as: "for a give sample, if the number of feature values that fall into the corresponding intervals of this rule exceeds 3 (which is the matching threshold, *r*), then…" These sets of rules can be converted into the following 3 rules:

1. ¬ Set 1 Λ Set 2 Λ Set 3 → *Setosa*
2. Set 1 Λ ¬ Set 2 Λ Set 3 → *Versicolor*
3. Set 1 Λ Set 2 Λ ¬ Set 3 → *Virginica*

where "Set *i*" stands for the given sample satisfy at least one rule in Set *i*, and "¬ Set *i*" stands for the given sample does not satisfy any rule in Set *i*.

6.3.3 Vulnerability Analysis

The proposed classifier has satisfying performance in the previous experiments. However, there is still room for performance improvement of the classifier. For the Iris dataset classifier, among the incorrectly classified patterns were patterns that could not be classified by the classifier, which were due to the false negatives made by one of the subsystems. For the Breast Cancer Dataset classifier, false negatives also led to malignant patterns classified as benign. Since GENERTIA is designed to effectively reduce the false negatives in an anomaly detection system, instinctively one will think of using GENERTIA to improve the performance of the classifier by reducing false positives.

In this section, vulnerability analysis is performed on the classification systems. A vulnerability is a pattern that is classified as normal by all the subsystems. Such a pattern cannot be classified by the classifier. In the pattern space, these patterns reside outside of the coverage of the detectors from all subsystems. The GRT can be employed to discover

these patterns, and the GBT can be used to design detectors for these vulnerabilities. This GRT/GBT cycles may effectively reduce the number of vulnerabilities in the system.

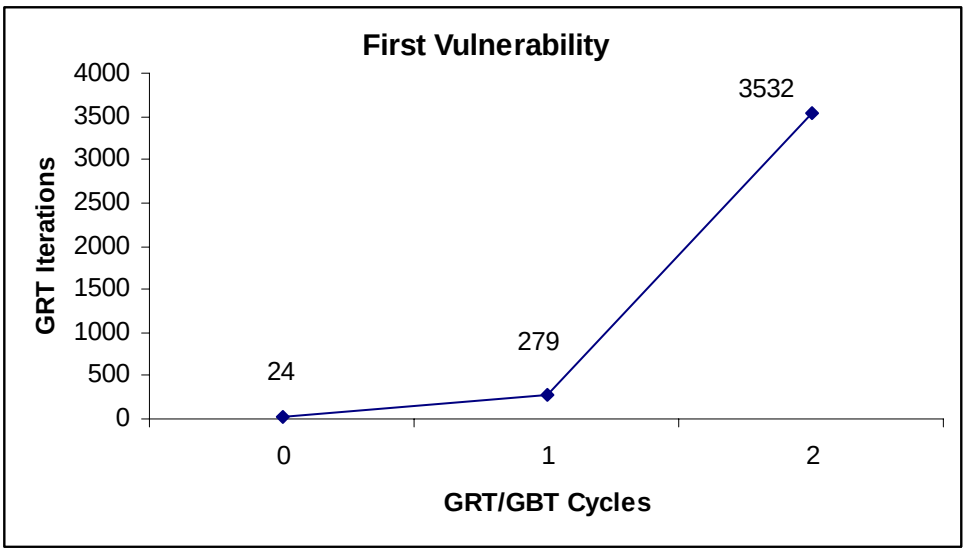
The detectors designed by the GBT should be added to the subsystems that should detect the corresponding patterns; otherwise, these detectors will cause false positives. For example, in the case of the Fisher's Iris dataset, if the GRT discovers a pattern that is not detected by all the three subsystems and this pattern actually represents a *Versicolor* iris, then detector that matches this pattern should be added to both *Setosa* and *Virginica* subsystems. However, which subsystems the detectors should be added to is not easily decided. GENERTIA has no way to make such a decision without the help from a domain expert. Unfortunately, domain experts were not available for this research[13]. In later experiments, these detectors are added to an additional set, and vulnerabilities discovered later are the samples that cannot be detected by all the detector sets. The purpose of these experiments is to show that the number of vulnerabilities of the subsystems in the Anomaly Detection-Based Classifier can be reduced by GENERTIA, which may improve the performance of the Classifier by reducing the number of patterns that cannot be classified. This is measured by the number of GRT iterations needed to find vulnerabilities in the classifier, because, as discussed in Chapter 5, the more GRT iteration needed the fewer vulnerabilities in the classifier.

[13] The American Iris Society and all the horticulture departments of the SEC universities have been contacted for assistance in iris pattern classification without success. We contacted the UAB medical school, and they couldn't help classify the breast cancer patterns.

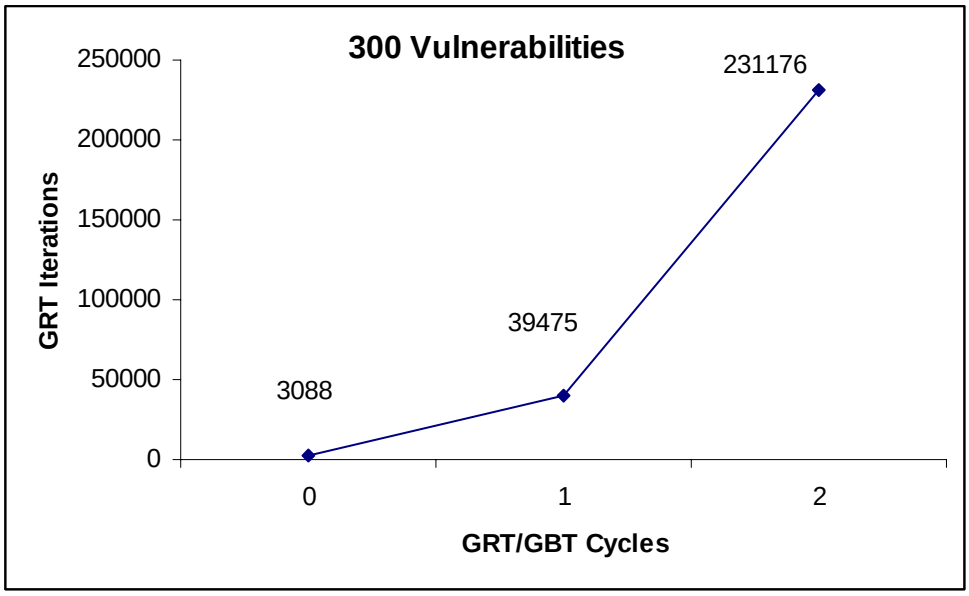
In this section, classifier for the Fisher's Iris Dataset consists of three subsystems and the classifier for the Wisconsin Breast Cancer Dataset consists of two subsystems, one for the benign patterns and one for the malignant patterns. These classifiers are trained with all the available patterns. Therefore the entire test set is now known to the classifier, and the classifier has 100% classification accuracy. Vulnerability analysis is performed on these 'perfect' classifiers[14]. First, a GRT was used to discover the 300 vulnerabilities in the system, and then a GBT was used to develop detectors that cover the holes discovered by the GRT. When these detectors were added to the system as an additional detector set, the vulnerabilities in the system is reduced — here we assume that with domain expert knowledge, the newly designed detectors can be deployed properly. The GRT/GBT cycle was carried out twice, and vulnerability analysis was carried out on the system. The GRT iterations needed to find the first and the 300th vulnerabilities were recorded to show the vulnerabilities in the classifier are reduced.

The experiment results shown in Figures 6.3 and 6.4 are the average of 10 runs for Fisher's Iris Dataset and Wisconsin Breast Cancer Dataset, respectively. The x-axis is the GRT/GBT cycles consumed for building/redesigning the system, and y-axis is the number of the GRT iterations needed to find the 1st (a) or the 300th (b) vulnerabilities in the classifiers. Cycle 0 marks the number of GRT iterations needed to find vulnerabilities

[14] Certainly training with entire dataset makes the classifiers overfit; however, this does not matter because our purpose is to show that GENERTIA can be employed to perform vulnerability analysis and system redesign of the classifier.

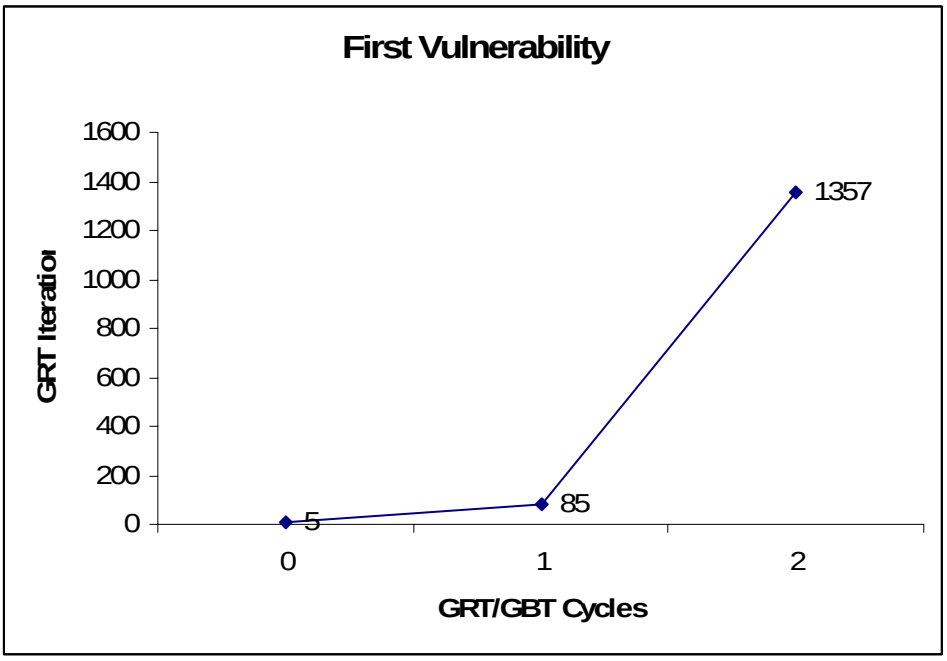


(a)

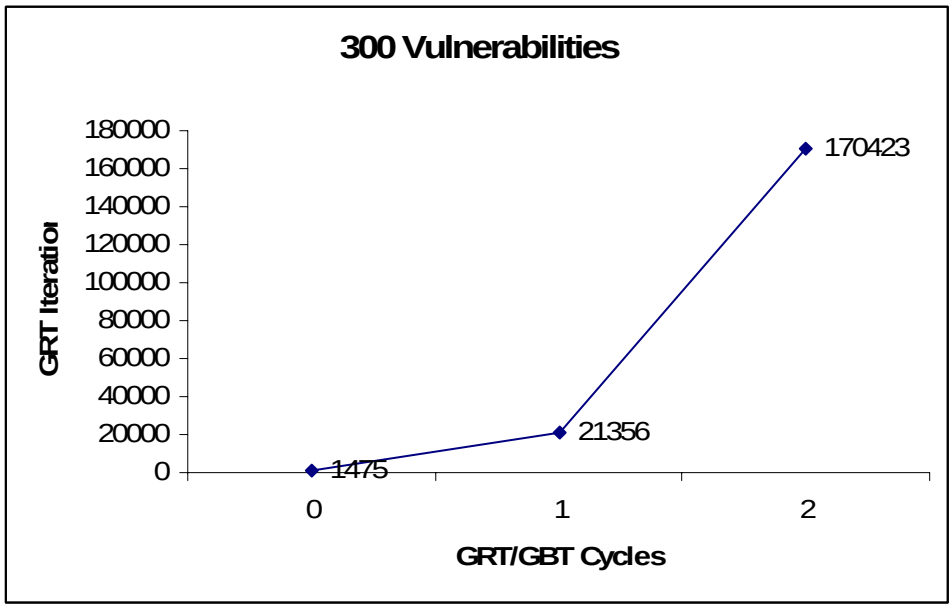


(b)

Figure 6.3 Vulnerability analysis of the classifier for Fisher's Iris Dataset. (a) The GRT effort needed to find 1 vulnerability (b) The GRT effort needed to find 300 vulnerabilities



(a)



(b)

Figure 6.4 Vulnerability analysis of the classifier for Wisconsin Breast Cancer Dataset. (a) The GRT effort needed to find the 1st vulnerability (b) the GRT effort needed to find the 300th vulnerability

in the initial system. Cycle 1 marks the number of GRT iterations needed to find vulnerabilities in the system after the additional detector set is added to the system. Cycle 2 marks the number of GRT iterations needed to find vulnerabilities in the system after a second GRT/GBT cycle.

The curves in these figures have the same shape. One can see from these figures that GENERTIA quickly reduces the vulnerabilities in the classifiers. During each GRT/GBT cycle, the average number of detectors added to the system is less than 3 for classifiers of both datasets. Assuming the GRT iterations needed to find vulnerabilities is proportional to the number of patterns the system can classify, the GRT/GBT cycles can reduce the number of patterns in an effective manner: adding a small number of detectors can exponentially reduce the number of patterns that cannot be classified by the classifier.

6.4 Discussion

In this Chapter, we propose a novel classification system based on a constraint-based anomaly detection system. A genetic algorithm that is employed by the GENERTIA blue team is used to generate very compact detector set that has high detection rate and low false positive rate as well. Experiment results showed that the classification accuracy of the system is favorably comparable to well-known classification systems. We also suggest that constraint-based detectors can be easily converted into rules for logical understanding the data being classified.

Although the classifier has satisfying performance on the Fisher's Iris Dataset and Wisconsin Breast Cancer Dataset, vulnerability analysis shows that the GRT can easily find patterns that cannot be classified by the classifier. The GBT can be employed to design detectors to cover the identified vulnerabilities. However, in order to properly deploy the newly designed detectors, domain expert knowledge is needed to classify the vulnerabilities.

Since the classifier is based on negative detectors, and as long as false negative exists, which is inevitable because the training set is not complete, there always will be patterns that cannot be classified. In the next chapter, we test the classifier using a third dataset and show that GENERTIA can be employed to improve the performance of a classifier with the help of a "virtual expert".

CHAPTER 7 APPLYING GENERTIA TO ANOMALY DETECTION-BASED CLASSIFIER

In this chapter, we apply GENERTIA to the proposed classification system to improve its classification accuracy. The accuracy of the proposed classifier depends on the performance of its subsystems. Patterns that are not correctly classified, including misclassified and "unclassified", are resulted from the two types of errors made by the subsystems: the false positives and false negatives. The GENERTIA cycles can be employed to reduce the number of "unclassified" samples by reducing the false negatives of the subsystems.

7.1 Introduction

An anomaly detection based classification system was presented in Chapter 6. The proposed classification system consists of a group of subsystems. For each subsystem, an AIS is built that contains a set of negative detectors, which can be generated through the Negative Selection Algorithm (NSA) or the GENERTIA blue team (GBT). As mentioned in Chapter 4, there are two types of errors associated with a hypothesis obtained with any learning algorithm, false positives and false negatives. These errors

will cause “misclassified” and / or “unclassified” errors in the proposed classification system. The errors of misclassification are usually fewer than errors resulted from “unclassified” samples. This is because a misclassification indicates that at least two subsystems simultaneously misclassify this pattern: one subsystem declares it abnormal but it actually does belong to this class, while another subsystem declares it normal but it actually does not belong to that class. From the experiment results in the previous chapter, we can see that the accuracy of the system is primarily decided by the number of unclassified patterns. So, reducing the number of unclassified patterns is important in order to improve the performance of the proposed classification system.

Both false positives and false negatives in a subsystem may lead to patterns being “unclassified”. Both false positives and false negatives are inevitable unless the training sets are complete but this is usually not true. The false negatives indicate the existence of vulnerabilities in the AIS. As discussed in previous chapters, simply increasing the size of the detector set using the NSA is not sufficient to reduce the number of vulnerabilities. One reason is that, because of the coverage overlaps among detectors, the gain in increased pattern space coverage is usually not proportional to the increase in the size of the detector. The other reason is that increase in the detector set also results in the increase in the false positive rate. A good learning algorithm should increase the

detection rate (because of increased pattern space coverage) with minimum increase in the false positive rate.

In this chapter, the proposed classification system is tested with a third data set, the Indian Pima Diabetes Dataset, and efforts are made to improve its accuracy.

7.2 The Pima Indian Diabetes Dataset

The Pima Indian Diabetes Dataset is a well-known benchmark dataset for machine learning research. A variety of classification algorithms have been tested on this dataset and no algorithm performed exceptionally well. Duch et al. [Duch 2002a] have collected the various past results on this dataset, as shown in Table 7.1. From this table we can see that the best of these results is at 77.7% with Logdisc.

The Pima Indian Diabetes dataset contains two classes of samples, healthy and diabetes. These samples have 8 attributes, including number of pregnancies, plasma glucose concentration in an oral glucose tolerance test, diastolic blood pressure (mm Hg), triceps skin fold thickness (mm), body mass index (kg/m^2), 2-hour serum insulin (mu U/ml), diabetes pedigree function, and age (years). Each sample is assigned a yes-or-no sign for diabetic test according to WHO criteria. Totally, there are 768 samples in the dataset, with 500 samples (65.1%) being healthy, and 268 samples (34.9%) being diabetes. Therefore, the best reported classification result on this dataset is just a little

more than 10% higher than the default accuracy (65.1%, which is obtained by classifying every sample as healthy), which means that this dataset presents a difficult classification problem.

Table 7.1 Some Previous Results for the Pima India Diabetes Dataset (Source: [Duch 2002a])

Method	Accuracy %
Logdisc (Logistic Discrimination)	77.7
DIPOL92	77.6
Linear Discriminative Analysis	77.5-77.2
SMART (Projection Pursuit Regression)	76.8
Fisher Discriminative Analysis	76.5
MLP+BP (Multi-Layer Perceptron + Back Propagation)	76.4
LVQ (Learning Vector Quantization)	75.8
RBF (radial basis function)	75.7
SSV DT (Separability Split Value Decision Tree)	73.7± 4.7
C4.5 DT (C4.5 Decision Tree)	73.0
Bayesian	72.2± 6.9
CART (Classification And Regression Trees)	72.8
SOM (Self-Organizing Map)	72.7
ID3 (Iterative Dichotomizer version 3)	71.7± 6.6
IBL (Instance Based Learning)	70.4± 6.2
kNN (k Nearest Neighbors)	67.6
C4.5 rules	67.0±2.9
OCN2 (Ordered CN2 induction algorithm)	65.1± 1.1
Default	65.1
QDA (Quadratic Discriminative Analysis)	59.5

All of the 8 attributes are continuous values, with each attribute having different data range. In order to be processed with the constraint-based anomaly detection system, this dataset was first discretized using following steps:

1. For each of the 8 attributes, the values from all the samples are collected and sorted.

2. The sorted values were equally divided into 40 sections.

3. Each section, defined as a value range with lower and upper values on the section boundaries, is assigned an integer starting from 0 in the ascending order.

Now, for a given sample, each value of its attributes can be mapped into a certain integer depending on the value range to which the value belongs. For example, for a given sample, if its value for the diabetes pedigree function is between 0.349 - 0.371, it will be assigned an integer of 18 for this attribute. Processed in this way, each attribute should have a data range of 0 – 39. However, for some attributes, there are more than 40 samples having the same value for a given attribute; therefore, the data ranges for these attributes were smaller than 39. The data ranges of the 8 attributes were 12, 39, 23, 28, 21, 39, 39, and 27 respectively. This means the size of the sample space is around 260 billion.

The choice of dividing the raw data into 40 sections was arbitrarily made, but it should be noted that this choice may affect the performance of the classifier. If a larger number is chosen, the sample space will be larger for the system to deal with, on the other hand, if a smaller number is chosen, the sample space will be smaller, but the normal samples and the abnormal samples may be closer to each other, blurring the difference between the two. It is also worth noting that, after the dataset is discretized this way, all the samples are distinct, which means that there is no sample occurring in both the normal and abnormal sets.

7.3 Initial Result: Classifiers without GENERTIA redesign

In this section, the two implementations of classifiers were tested: classifiers with detectors generated from random generation and negative selection (RGNS), and classifiers with detectors designed by the GBT. Since the dataset has only two classes, the classifiers can be implemented to have two subsystems, one for each class; the classifiers can also be implemented with one subsystem, and in this case, the classification problem is treated as an anomaly detection problem. Therefore, there were totally 4 test cases.

In the first test case, a classifier with a single subsystem of randomly generated detectors was tested. The healthy samples were used as normal and the detectors of the subsystem should detect diabetes samples as abnormal samples. 70% (350) of randomly

selected healthy samples were used for training and the rest 418 samples are used for testing. Since there are 8 attributes associated with each sample, there are 8 possible values for settings the matching threshold of the detector. All the 8 different settings of the matching threshold of the detectors were tested. The classifier was also tested with detector sets of different sizes. For each setting, the experiment was repeated for 10 times and the average results are shown in Tables 7.2 and 7.3. Table 7.2 shows the details of the classification errors, in terms of false negatives and false positives. Table 7.3 shows the corresponding accuracy, which is calculated as the number of errors divided by the number of samples in the testing set. From Table 7.3 we can see that increasing the size of the detector set resulted in higher accuracy, and from Table 7.2 we can see that the increase of accuracy was because of the decrease in the number of false negatives. The reason that the classifiers with higher detector matching thresholds had higher accuracy than the ones with lower matching thresholds was that the former ones were more general and tend to match more patterns, thus they had lower false negatives. However, increasing the size of the detectors also caused quick increase in the number of false positives, and that even with 6400 detectors the accuracy of the classifier was less than 60%, which was even lower than the default accuracy of 65%. The performance of this classifier was very poor.

Table 7.2 Errors Made by AIS with Randomly Generated Detectors
(with 70% healthy samples used for training and remaining samples (418) used for testing
The numbers in each cell are in the form of "false positives + false negatives", and the sum is the total errors)

	Size of Detector Set of the AIS in the Subsystem						
Matching Threshold	100	200	400	800	1600	3200	6400
3	168.8+54.2	134+79.1	103.8+101.6	77.1+119.5	60.2+130.7	42.8+136.6	33+139.4
4	199.9+39.9	169.1+60.2	138.4+78.5	115.4+93.8	97.3+110.8	77.9+120.1	66.2+130.8
5	226.8+22.2	209.3+36.7	180.4+50.2	160.5+68.1	140+81.9	120.8+95.9	106.8+107.9
6	252.8+9.6	241.1+14.8	226.9+25.6	210+40.4	191+54.4	174.8+65	158.3+76.9
7	263.8+3.5	261.9+5.2	256.7+8.7	250.5+13.1	241.6+22.5	231.5+29.4	215.6+41
8	267.9+0.1	267.2+0.5	257.3+7.2	265.8+1.5	264.1+3.1	262.2+4.2	257.2+8.2

Table 7.3 Accuracy of the AIS with Randomly Generated Detectors
(Accuracy calculated as the sum of false negatives and false positives divided by 418)

	Size of Detector Set of the AIS in the Subsystem						
Matching Threshold	100	200	400	800	1600	3200	6400
3	46.65%	49.02%	50.86%	52.97%	54.33%	57.08%	58.76%
4	42.63%	45.14%	48.11%	49.95%	50.22%	52.63%	52.87%
5	40.43%	41.15%	44.83%	45.31%	46.91%	48.16%	48.64%
6	37.22%	38.78%	39.59%	40.10%	41.29%	42.63%	43.73%
7	36.05%	36.10%	36.51%	36.94%	36.82%	37.58%	38.61%
8	35.89%	35.96%	36.72%	36.05%	36.08%	36.27%	36.51%

In the second test case, a classifier with a single subsystem consisting of detectors designed by the GBT was tested. Again, the healthy samples were used as normal and the diabetes samples as abnormal samples. 70% (350) of randomly selected healthy samples together with 70% (187) of randomly selected diabetes samples were used for the GBT to design detectors; the remaining 231 samples were used for testing. All the 8 possible settings of matching thresholds were all tested. Table 7.4 shows the average results of 10 runs for each experiment setup. As expected, detectors designed by the GBT had much better performance than the randomly generated detectors, for the same reason explained in Chapter 5. The accuracy of the classifiers, however, is only slightly higher than the default accuracy.

Table 7.4 Results for AIS with GBT-Designed Detectors
(with 70% samples used for training and 231 samples (30%) used for testing)

Matching Threshold	False Negatives	False Positives	Total Errors	Accuracy
3	29.0	55.0	84.0	63.64%
4	31.7	47.7	79.3	65.66%
5	39.7	37.7	77.3	66.52%
6	35.0	41.3	76.3	66.96%
7	38.0	36.7	74.7	67.68%
8	36.3	35.7	72.0	68.83%

In the third test case, a classifier with two subsystems of randomly generated detectors was tested. In this experiment, 70% of randomly selected samples from both classes were used for negative selection of detectors for the two subsystems respectively; the remaining 231 samples were used for testing. In this case, as related in Chapter 6, the correct classification of a sample requires that both subsystems correctly classify it as

normal or abnormal. There are three other possible situations that will result in an incorrect classification: 1) both subsystems wrongfully classify this sample, which results in a misclassification; 2) both subsystems classify it as normal, or 3) both classify it as abnormal. The sample is "unclassified" in the latter two situations. When an "unclassification" occurs, the classifier is forced to make a guess. Since the healthy samples clearly outnumber the diabetes samples (which means the healthy samples may have larger occupation in the sample space), an intuitive solution is to give bias to healthy when an unclassification takes place, in other words, to classify "unclassified" samples as healthy. Tables 7.5 and 7.6 show the average results of 10 runs for each experiment setup. Table 7.5 shows the details of the numbers of misclassification and unclassification, and Table 7.6 shows the result from the same experiment runs after bias is given to healthy for assigning class to unclassified samples.

In the fourth test case, the detectors of the two subsystems were designed by the GBT, which has the same implementation of the GBT as detailed in Chapter 6. Table 7.7 shows the average results of 10 runs for each experiment setup. Again, the detectors designed by the GBT had better performance than the randomly generated detectors.

Table 7.5 Results for 2-Subsystems Classifier with Randomly Generated Detectors
(with 70% samples used for training and 231 samples (30%) used for testing,
The numbers in the cells are in the form of "misclassified + unclassified")

	Size of Detector Set of the AIS in the Subsystem						
Matching Threshold	100	200	400	800	1600	3200	6400
3	28.0+125.5	19.5+148.4	11.9+174.7	5.0+204.2	2.8+216.3	1.5+224.0	0.2+229.1
4	27.1+136.5	23.4+133.0	19.0+158.4	13.8+177.8	7.9+192.4	3.1+210.9	0.8+226.5
5	23.1+157.9	25.6+148.1	21.7+153.6	20.2+160.1	13.3+171.4	6.6+194.7	2.3+216.2
6	14.0+193.4	22.5+174.9	20.2+169.1	23.0+164.7	23.9+167.7	18.2+169.4	10.5+191.5
7	5.3+217.4	10.0+209.6	13.0+199.1	16.1+192.4	17.9+181.8	19.3+178.2	17.6+179.6
8	0.3+230.2	0.2+229.7	1.2+228.2	1.7+227.6	3.5+221.1	5.8+217.8	12.9+198.2

Table 7.6 Results for 2-Subsystems Classifier with Randomly Generated Detectors
(with 70% samples used for training and 231 samples (30%) used for testing)

	Size of Detector Set of the AIS in the Subsystem						
Matching Threshold	100	200	400	800	1600	3200	6400
3	67.19%	65.80%	66.19%	66.41%	64.71%	65.79%	66.06%
4	66.06%	63.25%	64.48%	64.16%	64.07%	66.17%	64.35%
5	65.02%	60.13%	63.79%	60.17%	62.47%	63.42%	62.18%
6	64.55%	61.13%	62.18%	63.97%	62.64%	63.38%	62.35%
7	64.55%	63.33%	62.33%	62.73%	62.77%	64.48%	65.09%
8	64.98%	64.85%	64.37%	64.85%	64.85%	65.02%	65.44%

Table 7.7 Results for 2-Subsystems Classifier with GBT-Designed Detectors (with 70% samples used for training and 231 samples (30%) used for testing)

Matching Threshold	Error	Misclassified	Unclassified	Misclassified-by-Default	Total Misclassified	Accuracy
3	123.5	35.5	88.0	37.9	73.4	68.23%
4	119.2	37.4	81.8	34.7	72.1	68.79%
5	116.5	40.0	76.5	33.3	73.3	68.27%
6	114.0	35.1	78.9	35.1	70.2	69.61%
7	112.7	35.4	77.3	34.6	70.0	69.71%
8	114.9	36.2	78.7	34.6	70.8	69.35%

From these results, one can see that the performance of these classification systems were very poor, with the best performance obtained in the second test case having a slightly higher accuracy than the default value. These unsatisfying results may seem surprising, given the fact that, in the Chapter 6, the classifier had very good performance on the other two datasets. The main reason for this is the large overlapping between the normal (healthy) and abnormal (diabetes) sample space presented in this Dataset. While the proposed classification system is based on anomaly detection, the normal samples in this dataset are not very different from the abnormal samples.

There is also another reason for the unsatisfying performance of the system. From those tables, one can see that the classification errors were mainly caused by unclassification, a careful review of the results revealed that more than half of these unclassifications were due to both subsystems classifying a sample as normal, which means both subsystems had a lot of false negatives. As discussed in previous chapters, the existence of false negatives indicates the vulnerabilities in an anomaly detection

system. Since GENERTIA was developed to detect and cover the vulnerabilities in such a system, therefore, hopefully, GENERTIA can be applied to improve the performance of the classification system.

In the following sections, efforts were made to improve the classifier with two subsystems having detectors generated using the GBT. The reason whey we abandoned the other three implementations came with two concerns. First, the GBT-designed subsystems had better performance over the subsystems with randomly generated detectors, and second, with two subsystems in the classifier, we can focus on reducing the number of unclassification, since the two subsystems provides a crosscheck mechanism which results in fewer misclassifications.

7.4 Applying GENERTIA to the Classification System

The proposed classification system employs a GBT to design detectors using the known positive samples as abnormal patterns and the learning stops when all the known abnormal patterns are covered by the detector set. One major drawback of this detector generation approach is that, when the known positive samples are not representative of the whole positive pattern space, as is the case for the Pima India Diabetes Dataset, the resulting AIS may have large amount of vulnerabilities. The GENERTIA concept can be applied to the individual subsystems to improve the overall performance of the proposed classification system. After the individual subsystems are built, the

GENERTIA red team (GRT) and the GBT cycles can be applied to the AISs in the subsystems to reduce the vulnerabilities in the subsystems, and, hopefully, reduce the number of unclassifications.

A small modification has been made to the GBT. The original blue team tries to design detectors for every vulnerability, which means that the GBT returns only when all the vulnerabilities have been covered by the newly designed detectors. In the modified algorithm, the GBT is given a limit of certain number of iterations, and it exits if no valid candidate detector can be released in that limit. There are two concerns for this modification. First, since the normal and abnormal patterns have large overlap in the pattern space, there is a possibility that some abnormal patterns are so close to the normal patterns that detectors designed for these abnormal patterns will inevitably cover even more normal patterns, which results in more false positives than the number of false negatives that would be generated if these detectors were not designed. Second, since these abnormal patterns are so close to normal patterns, it is very difficult to design detectors for these abnormal patterns, and this modification prevents the GBT running for ever.

There are two strategies in redesign the classifier using GENERTIA, redesigning both subsystems in parallel or redesigning only one subsystem. Both strategies were

tested in this section. The reason for redesigning only one subsystem will be discussed later in this section.

With the first redesign strategy, GENERTIA was applied on both subsystems to reduce the number of false negatives in these two AISs. The experiments followed the following steps:

1. A classifier with two subsystems was built using the GBT, with 70% randomly selected health and diabetes samples as training sets.

2. The GRT was given a limit of 1,000,000 iterations to discover 100 vulnerabilities in one subsystem.

3. The GBT was used to design detectors for the subsystem that cover the vulnerabilities discovered in Step 2. The GBT breaks out if no valid detectors can be found in a limit of 1,000,000 iterations, with remaining vulnerabilities discarded.

4. Repeat Steps 2 – 3 for the other subsystem.

5. Repeat Steps 2 – 4 until one of the GRT couldn't find 100 vulnerabilities by the given iteration limit.

6. Test the classifier with the whole dataset, and whenever an unclassification occurred, the default value, "healthy", was assigned. Record the number of unclassifications and misclassifications.

Table 7.8 shows the experiment results. Compared with the results in Table 7.7, where the classifiers were not redesigned by GENERTIA, one can see that the performance of the classifiers surprisingly degraded after the GENERTIA redesign, with the accuracy dropped from around 69% to 65%. Careful reading the two tables, one can see that the reason was that the redesign introduced a lot of unclassified samples, although the number of misclassification reduced significantly. These samples were unclassified because both subsystems classified them as abnormal. In other words, both of the AISs in the two subsystems were making many false positives and around 85% of the samples were unclassified. These unclassified samples were assigned the default value "healthy", and the overall accuracy of the classifiers was very close to the default accuracy, 65%.

Table 7.8 Results for 2-Subsystems Classifier with 2 Subsystems Redesigned by GENERTIA (with 70% samples used for training and 231 samples (30%) used for testing)

Matching Threshold	Error	Misclassified	Unclassified	Misclassified-by-Default	Total Misclassified	Accuracy
3	216.1	5.6	210.5	75.8	81.4	64.76%
4	212.0	4.9	207.1	77.2	82.1	64.46%
5	212.7	10.6	202.1	70.7	81.3	64.81%
6	209.5	7.6	201.9	71.6	79.2	65.71%
7	209.8	9.2	200.6	71.9	81.1	64.59%
8	211.1	5.0	206.1	76.0	81.0	64.94%

With the second redesign strategy, GENERTIA was employed to reduce the number of false positives in only on AIS. In this case, the two subsystems are "unbalanced" in terms of the ability to detect anomalies, with the redesigned one being

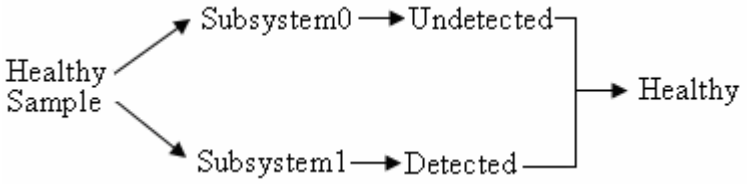
more powerful. The subsystem chosen for GENERTIA redesign was the subsystem that treated healthy samples as normal and diabetes samples as abnormal. The reason for this was based on the choice of the default value that will be assigned when an unclassification occurs: the classifier assigns "healthy" to unclassified samples. With the choice to redesign the subsystem that treated healthy samples as normal, it will be much safer to assign the default value to unclassified samples that are not detected by the redesigned subsystem. The idea is that, given an unclassified pattern, if it is not detected by both AISs of the two subsystems, then it is more likely that the weaker one caused the false negative; on the other hand, if the pattern is detected by both subsystems, then chances are that the weaker one is correct because the stronger one tends to make false positive errors. Therefore, when an sample can not be classified, if both subsystems detected it, it was classified as healthy because redesigned subsystem (the stronger one) probably has made a false positive; if both subsystems did not detect it, it is still classified as healthy because the subsystem that has not been redesigned (the weaker one) probably has made a false negative.

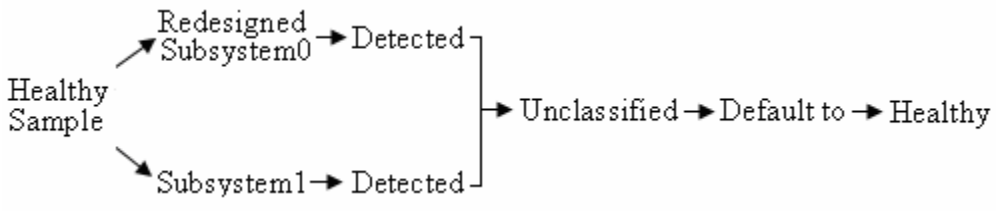
The following experiments followed the same steps listed in previous page except that only one subsystem was redesigned, the one that treat the healthy samples as abnormal. The results of these experiments are shown in Table 7.9.

Table 7.9 Results for 2-Subsystems Classifier with 1 Subsystem Redesigned by GENERTIA (with 70% samples used for training and 231 samples (30%) used for testing the redesigned subsystem was the one that treat diabetes samples as normal)

Matching Threshold	Error	Misclassified	Unclassified	Misclassified-by-Default	Total Misclassified	Accuracy
3	127.8	34.3	93.5	31.6	65.9	71.49%
4	124.8	35.3	90.5	31.6	66.9	71.06%
5	129.7	36.6	93.1	28.5	65.1	71.80%
6	125.4	33.8	91.6	30.9	64.7	72.01%
7	119.6	31.1	88.5	33.6	64.6	71.97%
8	121.3	32.7	88.6	33.7	66.4	71.24%

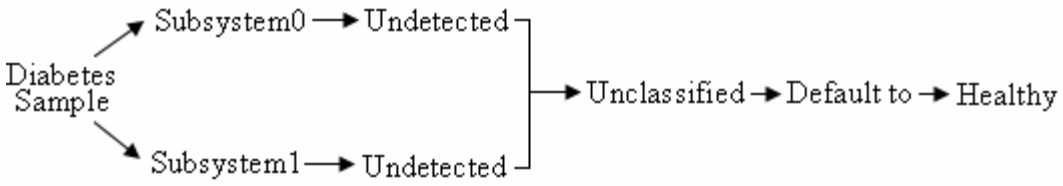
Compared with Table 7.7, one can see that the performances of the classifiers are clearly improved, with classification accuracy increased from around 69% to 71.5%. Careful review of these two tables reveals an interesting point. The redesigned classifiers had more errors (the total number of misclassified and unclassified samples) than the classifiers without redesign, and this is because more samples became unclassified. However, the redesigned classifiers made few misclassification errors, and fewer misclassification errors were made from the unclassified samples when they subjected to the default classification function. The GENERTIA redesign process had two effects. First, it increased the number of false positives, which resulted in the increase in the number of unclassifications; second, it reduced the number of false negatives made by the redesigned subsystem. The first effect is illustrated as follows:

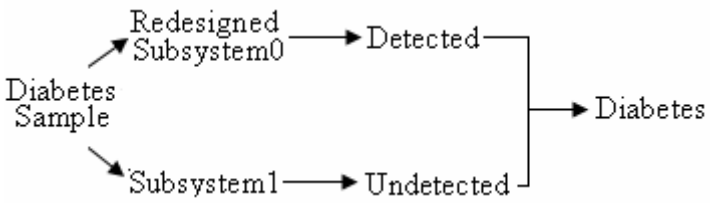




In this illustration, Subsystem0 is the subsystem that treats healthy samples as normal and Subsystem1 is the one that treats diabetes samples as normal. Here, a healthy sample is detected by Subsystem1 but not detected by Subsystem0, and then it is correctly classified. After the GENERTIA redesign process, Subsystem0 is able to detect this sample, which results in an unclassification and the sample is correctly classified by the default classification function.

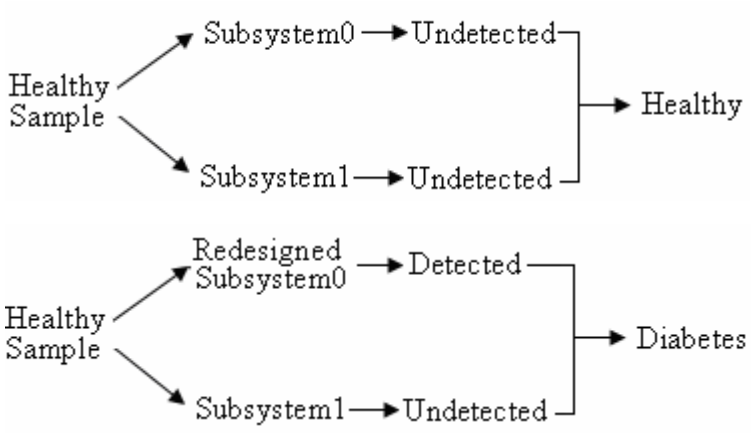
The second effect is illustrated as follows:





In this case, an unclassified sample that would be misclassified by default will now be correctly classified after the GENERTIA redesign process.

In the following case, however, the GENERTIA redesign process introduces false positives and negatively affects the performance of the classifier:



In this case, a false positive results in a misclassification. False positives are introduced when the vulnerabilities discovered by the GRT are actually unknown normal patterns and the GBT designs detectors for these patterns. Therefore, it is important to take into consideration of minimizing false positives when one uses GENERTIA to redesign the classifier.

7.5 A Virtual Expert

GENERTIA is originally developed to reduce the false negatives in anomaly detection systems. If a system has been redesigned to detect more anomalies, it also tends to generate more false positives. Different from the intrusion detection application domain as seen in the Chapter 5, where false negatives (attacks not detected) may be considered a more serious problem than false positives (normal traffics detected), both false positives

and false negatives are equally undesirable in a classification application. When applied to classification systems, GENERTIA needs some mechanism so that the false positives can be minimized.

To minimize the false positives, the candidate vulnerabilities discovered by the GRT should be confirmed to be real vulnerabilities (rather than some form of generalization of normal patterns) before they are used by the blue team to design detectors. Ideally, for a real classification application, this should be done by a domain expert. For example, in the case of the developing a classifier for predicting the presence of diabetes based on the 8 given covariates, a doctor is needed to decide the patterns, which the current classifier cannot correctly classified, represent healthy or diabetes. It should be noted that, adding the role of such a domain expert makes GENERTIA an interactive tool for design and/or redesign AISs for anomaly detection.

For this experiment, a virtual expert is developed to take the role of a doctor with expertise in predicting the presence of diabetes given a pattern that consists of the 8 attributes. A rule-based classifier is built based on the best reported rules discovered by other machine learning algorithms. Two rules, obtained from [Duch 2002], were employed to develop the rule-based classifier. These two rules are:

1. IF f2 > 144.5 OR (f2 > 123.5 AND f6 > 32.55) then diabetes, else healthy. This rule is from SSV, with an overall accuracy of 76.2%.

2. IF (f2 <=151 AND f6 <= 47) then healthy, else diabetes. This rule is from C-MLP2LN with L-units, with an overall accuracy of 75%.

In these two rules, f2 and f6 stands for the second and the sixth attributes of the samples. Combine these two rules, we have: "IF f2 > 144.5 OR (f2 > 123.5 AND f6 > 32.55) OR NOT (f2 <=151 AND f6 <= 47) then diabetes, else healthy", with an over accuracy of 76.4%. After converting the numbers in the rule to their corresponding integers according to intervals obtained in the discretization process, the rule becomes "IF (f2 > 22 OR f2 > 17 AND f6 > 16 OR NOT (f2 < 23 AND f6 < 29)) then diabetes, else healthy" with an over accuracy of 76.4%.

A GA was used to optimize these parameters, and we obtained a number of rules with high accuracy. From these obtained rules, one was selected and simplified as "IF (f2 > 25 AND f6 > 15 OR f2 >= 35 OR f6 >= 39) then diabetes, else healthy", which has an accuracy of 78%, better than the best result obtained from all known classifiers shown in Table 7.1. Converted back to the original measurements, this rule becomes "IF (f2 > 127 AND f6 > 29.9 OR f2 >= 160 OR f6 >= 49) then diabetes, else healthy". The confusion matrix of this virtual expert rule is:

	Healthy	Diabetes
Healthy	435	104
Diabetes	65	164

From this confusion matrix chart, one can see that the virtual expert tends to misclassify diabetes as healthy and it has high accuracy in predicting healthy samples. This means the vulnerabilities confirmed by this virtual expert are very likely being real vulnerabilities. Although this also means that some real vulnerabilities discovered by the GRT may be disconfirmed by the virtual expert, it is not so harmful to the redesign process as long as the confirmed vulnerabilities are real ones. This confusion matrix also suggests that the disconfirmed vulnerabilities should not be regarded as normal patterns and added to the normal set, because the probability of these patterns being abnormal is somehow high.

Although the accuracy of this virtual expert is far from being perfect, it is better than the best known classifiers for the Pima Indian Diabetes Dataset. Anyway, having this virtual expert checking the discovered vulnerabilities is better than not having one at all. In the following section, this virtual expert is used to check the vulnerabilities discovered by the GRT, and only those confirmed ones are given to the GBT to design detectors.

7.6 GENERTIA Redesign with Virtual Expert

The experiments conducted in this section were the same as the ones in Section 7.4 where GENERTIA was applied to redesign only one subsystem, except that an additional step was inserted into the GRT/GBT cycles: the vulnerabilities discovered by the GRT were

checked by the virtual expert, and only those that were confirmed by the virtual expert to by real vulnerabilities were given to the GBT to design corresponding detectors, and the rest were simply discarded.

For each parameter setup, the experiment was repeated 10 times and Table 7.10 shows the average results of these experiments. Comparing Tables 7.10 and 7.9, one can see that the virtual expert had improved the performance of the classifiers. Both misclassified and unclassified samples were fewer than the corresponding errors made by classifiers redesigned without the help of the virtual expert.

Table 7.10 Results for Classifier Redesigned by GENERTIA with Virtual Expert (with 70% samples used for training and 231 samples (30%) used for testing the redesigned subsystem was the one that treat diabetes samples as normal)

Matching Threshold	Error	Misclassified	Unclassified	Misclassified-by-Default	Total Misclassified	Accuracy
3	113.8	31.4	82.4	29.6	61.0	73.59%
4	109.1	32.1	77.0	27.8	59.9	74.09%
5	110.2	30.6	79.6	31.5	62.1	73.13%
6	106.4	27.8	78.6	32.0	59.8	74.11%
7	104.6	28.2	76.4	31.3	59.5	74.23%
8	103.3	30.5	72.8	29.9	60.4	73.86%

7.7. Summary

In Chapter 5, GENERTIA was applied to improve the performance of an AIS-based intrusion detection system. In this chapter, the GENERTIA was applied to another application, the proposed anomaly detection-based classification system. The GRT can be applied to discover the vulnerabilities in the classification system, specifically, the

GRT can be used to identify the patterns that are classified to be 'self' by more than two subsystems. If we can assume that each pattern can only belongs to one class, then these patterns represent vulnerabilities in one or more subsystems. Domain expert knowledge is needed to confirm the actual classes that these patterns belong to, and then the GBT can be used to design detectors for these patterns. The proposed classifier had poor performance on a third dataset, the Pima Indian Diabetes Dataset. Experiments showed that the performance of the classifier can be improved with the GENERTIA redesign process.

CHAPTER 8 CONCLUSION AND FUTURE WORK

8.1 Research Objectives

The goal of this dissertation is to improve the performance of an Artificial Immune System for anomaly detection. Typically, an AIS uses the Negative Selection Algorithm to generate negative detectors, which are capable of detect patterns possessing characteristics different from the normal patterns that are used for negative selection. An AIS can be described with three factors: the representation of the self/nonself patterns and the detectors, the matching rule that decides how patterns are matched by detectors, and an algorithm that describes how the detectors are generated, killed and regenerated. Accordingly, the work in this dissertation was focused around these three factors, and the objectives of this dissertation include: to propose new representations for self/nonself patterns and detectors, to propose new matching rules, and to propose new algorithms for generating detectors.

8.2 Dissertation Summary

New representations for self/nonself patterns and detectors have been proposed in this research. AISs based on constraint-based detectors and corresponding matching rules,

including Any-r-interval matching and Contiguous-r-interval matching rules have being implemented. With the new representations, self/nonself patterns are in the form of vectors of integers, and detectors are in the form of vectors of intervals. This high-level representation facilitates the abstraction of meaningful knowledge.

GENERTIA, a system used for AIS vulnerability analysis and redesign has been implemented. GENERTIA contains two parts, a red team that is used to discover vulnerabilities in an AIS, and a blue team that is used to design valid detectors for the discovered vulnerabilities. These two teams cooperate in a co-evolutionary fashion to redesign the AIS.

To investigate the effectiveness of GENERTIA, it has been applied to two AIS-based systems, including a network intrusion detection system (IDS), and an AIS-based classification system. Experiments show that GENERTIA can be used to improve the performance of AIS-based IDSs in terms of both efficiency and effectiveness.

The GENERTIA blue team provides a novel approach to the generation of detectors. This leads to the construction of an AIS-based classification system. This classifier consists of multiple subsystems in the form of AISs, one for each class of samples in the classification problem. The classification of a sample is decided by the combined results from all the AISs. The performance of this classification system has

been tested using two benchmark testing datasets, including the Wisconsin Breast Cancer Dataset and the Fisher's Iris Dataset. Results show that the proposed classification system is favorably comparable to some well-known classification systems.

The proposed classifier performed poorly on a third dataset, the Pima India Diabetes Dataset. One main reason was that the subsystems made many false negatives, which caused the classifier unable to classify the corresponding samples. GENERTIA can be used to reduce the vulnerabilities (which may result in false negatives) in individual subsystems of the classification system. However, the GENERTIA redesign process also introduces more false positives, which also causes inaccurate classification. This is because some vulnerabilities actually may be unknown normal samples rather than abnormal samples. To minimize the false positives, the vulnerabilities discovered by the GENERTIA red team needs to be confirmed by domain experts before the blue team design detectors for these vulnerabilities. This makes GENERTIA an interactive tool for redesign AIS-based anomaly detection system. Experiment shows the application of GENERTIA to the classifier improved its performance on the Diabetes dataset.

8.3 Future Work

There are several directions for continuing research efforts. First, it is important to be able to mathematically calculate the efficiency and effectiveness of constraint-based AISs,

in terms of the theoretical effort needed to generated a certain number of detectors, the theoretical possibility of a pattern being or not being detected by a detector set of given size. This work has been done for the binary-coded version.

The AISs implemented in this dissertation make a crisp boundary between the self and nonself space, which is based on the assumption that the normal is distinct from abnormal. However, for some application domain, it is possible that there is no such a clear cut between the two spaces. It may be interesting to build "fuzzy-version" constraint-based AISs. One possible approach involves the building of AISs with different levels of coverage, which detectors at each level having a certain matching threshold. Matching threshold defines the generality of specificity of the matching between the detectors and the patterns, and detectors with higher matching threshold are more specific than detectors with lower matching threshold. Therefore, with a set of detector with different matching thresholds, once an abnormal pattern is matched, it will be possible to tell at which level the pattern is matched by the detector set, and matching by detectors with higher matching thresholds indicates a higher possibility of the matched pattern being abnormal.

The GENERTIA blue team uses a GA to design detectors for the discovered vulnerabilities. The candidate detectors are assigned a fitness value of zero if they match any sample in the training set. While this works fine for datasets in which the normal and

abnormal samples are distinct, for dataset such like the Pima India Diabetes Dataset, where the two classes of data have large overlap in pattern space, it may be more effective to adopt a different candidate detector fitness function. For example, rather than assigning zero to candidates that cover training samples, the fitness of the candidates are given a penalty for each training samples they cover. This may generate detectors that have large coverage of abnormal space and only a little normal space.

Another interesting research direction is to abstract rules from the proposed classification system. Since it is based on constraint-based detectors, the detectors are already in the form of M-of-N rules. However, it will be more useful if these rules can be simplified in a form that can be directly understood by man. For complex dataset, such like the Pima India Dataset, the detector set becomes much larger, this simplification is more necessary. This may require the dissection of detectors in to individual intervals and ranking them according to the number of samples they matches for different classes.

These ideas, and others, extends beyond the research reported in this dissertation, and might be direction for future research.

REFERENCES:

[Anderson et al. 1995] Anderson, D., Frivold, T., and Valdes, A. "Next-generation intrusion detection expert system (NIDES): a summary", *Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International*, 1995, available at: http://www.sdl.sri.com/papers/4/s/4sri/ 4sri.pdf.

[Aicklin and Cayzer 2002] Aicklin, U. and Cayzer, S. "The danger theory and its application to artificial immune systems", *Proceedings of the First International Conference on Artificial Immune System*, 141-148.

[Ayara et al. 2002] Ayara, M., Timmis, J, de Lemos, R., and Duncan, R. "Negative Selection: How to Generate Detectors", *Proceedings of the First International Conference on Artificial Immune System*, 89-98.

[Balthrop et al. 2002a] Balthrop, J., Forrest, S. and Glickman, M. "Revisiting LISYS: Parameters and Normal Behavior", *Proceedings of the 2002 Congress on Evolutionary Computation*, 1045-1050.

[Balthrop et al. 2002b] Balthrop, J., Esponda, F., Forrest, S., Glickman, M. "Coverage and generalization in an Artificial Immune System", *Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, 3-10.

[Cooke and Hunt 1995] Cooke, D. and Hunt, J. "Recognizing Promoter Sequences Using an Artificial Immune System", in *Proceedings of the Intelligent Systems in Molecular Biology Conference*, 89-97, AAAI Press, 1995.

[Dasgupta and Forrest 1995] Dasgupta, D. and Forrest, S. "Tool breakage detection in milling operations using a negative selection algorithm", *Technical Report CS95-5*, *University of New Mexico*.

[Dasgupta and Forrest 1996] Dasgupta, D. and Forrest, S. "Novelty detection in time series data using ideas from immunology", *Proceedings of the ISCA 5th International Conference on Intelligent Systems*.

[Dasgupta 1999] Dasgupta, D. "An overview of artificial immune system and their applications", *Artificial Immune Systems and Their Applications*, D. Dasgupta, ed., Springer, 3-21, 1999.

[Dasgupta and Gonzalez 2002] Dasgupta, D. and Gonzalez, F. "An immunity-based technique to characterize intrusions in computer network", IEEE Transactions on Evolutionary Computation. 6(3):281-291.

[Dasgupta et al. 2004] Dasgupta, D., KrishnaKumar, K., Wong, D. and Berry, M. "Negative selection algorithm for aircraft fault detection", *Proceedings of the 2004 International Conference on Artificial Immune Systems*, 1-13.

[De Castro and Timmis 2002a] de Castro, L. and Timmis, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, ISBN 1852335947.

[De Castro and Timmis 2002b] de Castro, L. and Timmis, J. "An Artificial Immune Network for Multimodal Function Optimization", *Proceedings of the World Congress of Computational Intelligence*.

[De Castro and Timmis 2003] de Castro, L. and Timmis, J. "Artificial immune systems as a novel soft computing paradigm", *Soft Computing* 7:526-544.

[De Castro and Von Zuben 2000] de Castro. L. and Zuben, F. "An evolutionary immune network for data clustering", *Proceedings of the Brazilian Symposium on Artificial Neural Networks*, 84-89.

[De Castro and Von Zuben 2004] de Castro. L. and Zuben, F. *Recent Developments in Biologically Inspired Computing*, Idea Group Publishing, 2004.

[DeJong and Spears 1993] DeJong, K. and Spears, W. "On the state of Evolutionary Computation", *Proceedings of the fifth International Conference of Genetic Algorithm*, 618-623.

[D'haeseleer et al. 1996] D'haeseleer, P., Forrest, S. and Helman, P., "An immunological approach to change detection: algorithms, analysis and implications", *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, 110-119.

[Dozier 2003] Dozier, G. "IDS vulnerability analysis using GENERTIA red teams", *Proceedings of the International Conference on Security and Management*, 2003, pp. 171-176.

[Dozier et al. 2004b] Dozier, G., Brown, D., Hurley, J., and Cain, K. "Vulnerability analysis of AIS-based intrusion detection systems via genetic and particle swarm red teams", *Proceedings of the 2004 Congress of Evolutionary Computation*, pp. 111 – 116.

[Dozier et al. 2004a] Dozier, G., Brown, D., Hurley, J., and Cain, K. "Vulnerability analysis of immunity-based intrusion detection systems using evolutionary hackers",

Proceedings of the 2004 Genetic and Evolutionary Computation Conference, pp. 263 – 274.

[Duch 2002a] Duch, W. Datasets Used for Classification: Comparison of Results, available at: http://www.phys.uni.torun.pl/kmk/projects/datasets.html

[Duch 2002b] Duch, W. Logical Rules Extracted from Data, available at: http://www.phys.uni.torun.pl/kmk/projects/rules.html

[Duch et al. 2001] Duch W., Adamczak R. and Grabczewski K. "A new methodology of extraction, optimization and application of crisp and fuzzy logical rules", *IEEE Transactions on Neural Networks*, 12:277-306.

[Dunham 2003] Dunham, M. Classification. *Data Mining, Introductory and Advanced Topics.* Prentice Hall, Upper Saddle River, NJ, 2003.

[Esponda et al. 2004] Esponda, F., Forrest, S., and Helman, P. "A formal framework for positive and negative detection schemes", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 34(1):357-373.

[Farmer et al. 1986] Farmer, J., Packard, N. and Perelson, A. "The Immune System, Adaptation, and Machine Learning", *Physica D*, 22:187-204.

[Forrest et al. 1993] Forrest, S., Javornik, B., Smith, R. and Perelson, S. "Using genetic algorithms to explore pattern recognition in the immune system", *Evolutionary Computation*, 1:191-211.

[Forrest et al. 1994] Forrest, S., Perelson, S., Allen, L. and Cherukuri, R. "Self-nonself discrimination in a computer", *Proceeding of the 1994 IEEE Symposium on Research in Security and Privacy*, 202-212.

[Forrest et al. 1996] Forrest, S., Hofmeyr, S., Somayaji, A., and Longstaff, T. "A sense of self for unix processes", *Proceeding s of the IEEE symposium on Security and Privacy*, 1996, pp. 120-128.

[Forrest et al. 1997] Forrest, S., Hofmeyr, S. and Somayaji. A. "Computer immunology", *Communications of the ACM*, 40(10):88-96.

[Goldberg and Deb 1989] Goldberg, D. and Deb, K. "A comparative analysis of selection schemes used in genetic algorithms", *Foundations of Genetic Algorithms*, Rawlins, G. ed., pp. 69-93, Morgan Kaufmann, 1989.

[Gonzalez et al. 2002] Gonzalez, F., Dasgupta, D., and Kozma, R. "Combining negative selection and classification techniques for anomaly detection", *Proceeding of the 2002 Congress on Evolutionary Computation*, 2002, pp. 705-710.

[Gonzalez 2003] Gonzalez, F. "A study of artificial immune systems applied to anomaly detection", PhD dissertation, Department of Computer Science, University of Memphis.

[Gonzalez et al. 2003a] Gonzalez, F., Dasgupta, D., and Gomez, J. "The effect of binary matching rules in negative selection", *Proceedings of the 2003Genetic and Evolutionary Computation Conference*.

[Gonzalez et al. 2003b] Gonzalez, F, Dasgupta, D. and Nino, D. "A randomized real-valued negative selection algorithm", *Proceedings of the 2003 International Conference on Artificial Immune Systems*, 261-272

[Hamaker and Boggess 2004] Hamaker, J. and Boggess, L. Non-Euclidean Distance measures in AIRS, and Artificial Immune Classification System, *Proceedings of the* 2004 *IEEE conference on Evolutionary Computation.* 1067-1073.

[Harmer et al. 2002] Harmer, P., Williams, P., Gunsch, G., and Lamont, G. "An artificial immune system architecture for computer security applications", *IEEE Transactions on Evolutionary Computation*, 6(3):252-280.

[Hart and Timmis 2005] Hart, E. and Timmis, J. "Application areas of AIS: the past, the present and the future", *Proceedings of the 2005 International Conference of Artificial Immune System*, 483-497.

[Hastie et al. 2001] Hastie, T., Tibshirani, R., and Friedman J. eds. *The Element of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics). New York: Springer-Verlag, 2001.

[Haykin 1999] Haykin, S. *Neural Networks: A Comprehensive Foundation*, Prentice Hall, ISBN 0-13-273350-1

[Heberlein et al. 1990] Heberlein, T., Dias, V., Levitt, N., Mukherjee, B., Wood, J. and Wolber, D. "A network security monitor", *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pp. 296-303.

[Hofmeyr et al. 1998] Hofmeyr, S., Forrest, S. and Somayaji, A. "Intrusion Detection Using Sequences of System Calls", *Journal of Computer Security*, 6:151-180.

[Hofmeyr 1999] Hofmeyr (1999) "An immunological model of distributed detection and its application to computer security", PhD dissertation, Department of Computer Science, University of New Mexico.

[Hofmeyr and Forrest 2000] Hofmeyr, S., and Forrest, S. "Architecture for an artificial immune system", *Evolutionary Computation*, 8(4):443-473.

[Holland and Reitman 1978] Holland, J. and Reitman, J. "Cognitive Systems Based on Adaptive Algorithms", in Watermand, D. and Hayes-Roth, F. editors, *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

[Hou et al. 2002] Hou, H., Zhu, J., and Dozier, G. "Artificial immunity using constraint-based detectors", *Proceedings of the 2002 World Automation Congress*, pp. 239-244.

[Hou and Dozier 2004] Hou, H., and Dozier, G. "Comparing the performance of binary-coded detector and constraint-based detector", *Proceedings of the 2004 Congress of Evolutionary Computation*, pp. 773-777.

[Hou and Dozier 2005] Hou, H. and Dozier, G. "Immunity-Based Intrusion Detection System Design, Vulnerability Analysis, and GENERTIA's Genetic Arms Race", *Proceedings of the ACM Symposium on Applied Computing*, 961-965.

[Hou and Dozier 2006a] Hou, H. and Dozier, G. "An Anomaly-Detection Based Classification System", *Proceedings of the Congress of Evolutionary Computation*, CEC2006.

[Hou and Dozier 2006b] Hou, H. and Dozier, G. "An Evaluation of Negative Selection Algorithm with Constraint-Based Detectors", *Proceedings of 44th ACM Southeast Conference*, 134-139

[Janeway and Travers 2001] Janeway, C. and Travers, P. *Immunobiology: The Immune System in Health and Disease*, Current Biology Ltd., London, 2001

[Jerne 1974] Jerne, N. "Towards a Network Theory of the Immune System", *Annul Review of Immunology*, 125C:373-389, 1974.

[Ji and Dasgupta 2004a] Ji, Z. and Dasgupta, D. "Real-valued negative selection algorithm with variable-sized detectors", *GECCO 2004, Lecture Note in Computer Science*, 3102:287-298.

[Ji and Dasgupta 2004b] Ji, Z. and Dasgupta, D. Augmented Negative Selection Algorithm with Variable-Coverage Detectors, *Proceedings of the 2004 Congress on Evolutionary Computation*, 1080-1088.

[Ji and Dasgupta 2005] Ji, Z. and Dasgupta, D. "Estimating the detector coverage in a negative selection algorithm", *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, ???-???.

[Kabay 2001] Kabay, M. "Studies and surveys of computer crime", at: http://www.securitystats.com/reports/Studies_and_Surveys_of_Computer_ Crime.pdf.

[Kennedy et al. 1998] Kennedy, R., Lee, Y., Roy, B., Reed, C., and Lippman, R. *Solving Data Mining Problems through Pattern Recognition*, Englewood Cliffs, N.J.: Prentice Hall, 1998.

[Kim and Bentley 2001a] Kim, J. and Bentley, P.J., "An evaluation of negative selection in an artificial immune system for network intrusion detection", *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 1330-1337.

[Kim and Bentley 2001b] Kim, J. and Bentley, P. "Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator", *Proceedings of the 2001 Congress on Evolutionary Computation*, 1244-1252.

[Kim and Bentley 2004] Kim, J. and Bentley, P. J. "Immune memory and gene library evolution in the dynamic clonal selection algorithm", *Genetic Programming and Evolvable Machine*, 5(4):361-391.

[Krogh 2000] Krogh, D. "The immune system: defending the body from invaders", *Biology*, ISBN: 0023668911, 521-532.

[Langman and Cohn 2000] Langman, R. and Cohn, M. "Editorial summary", *Seminars in Immunology*, 12:343-344.

[Lee and Sim 2004] Lee, D. and Sim, K. "Negative Selection Algorithm for DNA Sequence Classification" *Proceedings of 2nd International Conference on Soft Computing and Intelligent Systems*.

[Leon 2000] Leon, M. "Internet virus boom", *Infoworld*, 22(3): 36-37, 2000.

[Lincoln Lab 1998] Lincoln Laboratory, 1998 DARPA Intrusion Detection Evaluation Data Set, at:http://www.ll.mit.edu/IST /ideval/data/1998/1998_data_index.html.

[Marchette 2001] Marchette, D.J. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*, Springer, 2001.

[McHugh 2000] McHugh, J. "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluation as performed by Lincoln Laboratory", *ACM Transaction*, 3(4):262-294, 2000.

[Mitchell 1997] Mitchell, T. *Machine Learning*, McGraw-Hill, 1997.

[Mukherjee et al. 1994] Mukherjee, B., Heberlein, T., and Levitt, K. "Network Intrusion Detection", *IEEE Network*, 8(3):26-41.

[Orr 1996] Orr, M. Introduction to radial basis function networks, *Tech. Rep., Centre for Cognitive Science, University of Edinburgh*, 1996

[Percus et al. 1993] Percus, J., Percus, O. and Perelson, A. "Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination", *Proceedings of the National Academe of Sciences, U.S.A.*, 90:1691-1695.

[Potter and De Jong 1998] Potter, M. and De Jong, K. "The coevolution of antibodies for concept learning". *Lecture Notes in Computer Science*, 1498: 530–535.

[Rogers and Tanimoto 1960] Rogers, S. and Tanimoto, T. "A computer program for classifying plants." *Science*, 132:1115-1118.

[Quinlan 1986] Quinlan, J. Induction of decision trees, *Machine Learning*, 11(1):81-106.

[Singh 2002] Singh, S. "Anomaly detection using negative selection based on the r-contiguous matching rule", *Proceedings of the 2002 International Conference on Artificial Immune Systems*, 99-106.

[Stepney et al. 2004] Stepney, S., Smith, R., Timmis, J. and Tyrell, A. "Towards a Conceptual Framework for Artificial Immune Systems", *Proceedings of the 3rd International Conference on Artificial Immune Systems*, 53-64.

[Stibor et al. 2004] Stibor, T., Bayarou, K. and Eckert, C. "An investigation of r-chunk detector generation on higher alphabets", *GECCO 2004, Lecture Note in Computer Science*, 3102:299-307.

[Stibor et al. 2005a] Stibor, T., Timmis, J. and Eckert, C. "On the appropriateness of negative selection defined over hamming shape-space as a network intrusion detection system", *Proceedings of the 2005 Congress of Evolutionary Computation*, 995-1002.

[Stibor et al. 2005b] Stibor, T., Timmis, J. and Eckert, C. "A comparative study of real-valued negative selection to statistical anomaly detection techniques", *ICARIS 2005, Lecture Note in Computer Science*, 3627:262-275.

[Stibor et al. 2005c] Stibor, T., Mohr, P. and Timmis, J. "Is negative selection appropriate for anomaly detection?", *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pp. 321-328.

[Sywerda 1989] Sywerda, G. "Uniform crossover in genetic algorithm", *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 2-9.

[Tanase 2002] Tanase, M. "One of these things is not like the others: the state of anomaly detection", 2002, available at: http://online.securityfocus.com/infocus/1600.

[Timmis et al. 2000] Timmis, J., Neal, M. and Hunt, J. "An Artificial Immune System for Data Analysis", *Biosystems*, 55(1): 143-150, 2000.

[Timmis and Neal 2000] Timmis, J. and Neal, M. "A Resource Limited Artificial Immune System for Data Analysis", in *Research and Development in Intelligent Systems XVII, Proceedings of ES2000*, (Canbridge, UK), 19-32, 2000.

[Timmis et al. 2004] Timmis, J., Knight, T., de Castro, L. and Hart, E. "An overview of artificial immune systems", Paton, R., Bolouri, H., Holcombe, M. Parish, J. and Tateson, R. editors, *Computation in Cells and Tissues: Perspectives and Tools for Thought*, Natural Computation Series, 51-86. Springer, November 2004.

[Taylor and Corne 2003] Taylor, D. and Corne, D. “An investigation of the negative selection algorithm for fault detection in refrigeration systems”, *Proceedings of ICARIS 2003*.
[Watkins and Boggess 2002] Watkins, A. and Boggess, L. “A resource limited artificial immune classifier”, *Proceedings of the 2002 Congress on Evolutionary Computation*, 1546-1551.

[Watkins and Timmis 2002] Watkins, A and Timmis, J. “Artificial Immune Recognition System(AIRS): Revisions and Refinements”, *Proceedings of the 1st International Conference on Artificial Immune Systems*, 173-191, 2002.

[Watkins et al. 2003] Watkins, A., Timmins, J. and Boggess, L. “Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm”, *Genetic Programming and Evolvable Machines*, 5(3): 291-317, 2004

[Wierzchon 2000] Wierzchon, S. “Generating optimal repertoire of antibody strings in an artificial immune system”, *Intelligent Information Systems, Advances in Soft Computing Series of Physica-Verlag*, Springer Verlag, 119-133.
[Witten and Frank 2000] Witten, I. and Frank, E. *Data Mining Practical Machine Learning Tools and Techniques*, San Francisco: Morgan Kaufmann, 2000.

[Wolberg and Mangasarian 1990] Wolberg, W. and Mangasarian, O. “Multisurface method of pattern separation for medical diagnosis applied to breast cytology”, *Proceedings of the National Academe of Sciences, U.S.A.*, 87:9193-9196.