The Artificial Immune System with Evolved Lymphocytes

by

Alexander J. Graaff

Submitted in partial fulfillment of the requirements for the degree

Magister Scientia

in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

November 2003

Abstract

The main purpose of the natural immune system is to protect the body against any unwanted foreign cells that could infect the body and lead to devastating results. The natural immune system has different lymphocytes to detect and destroy these unwanted foreign patterns. The natural immune system can be modeled into an artificial immune system that can be used to detect any unwanted patterns in a non-biological environment. One of the main tasks of an immune system is to learn the structure of these unwanted patterns for a faster response to future foreign patterns with the same or similar structure. The artificial immune system (AIS) can therefore be seen as a pattern recognition system. The AIS contains artificial lymphocytes (ALC) that classify any pattern either as part of a predetermined set of patterns or not. In the immune system, lymphocytes have different states: Immature, Mature, Memory or Annihilated. Lymphocytes in the annihilated state needs to be removed from the active set of ALCs. The process of moving from one state to the next needs to be controlled in an efficient manner. This dissertation presents an AIS for detection of unwanted patterns with a dynamical active set of ALCs and proposes a threshold function to determine the state of an ALC. The AIS in the dissertation uses evolutionary computation techniques to evolve an optimal set of lymphocytes for better detection of unwanted patterns and removes ALCs in the annihilated state from the active set of ALCs.

Supervisor: Prof. A.P. Engelbrecht
Department of Computer Science
Degree: Magister Scientiae

KEYWORDS:

Self, non-self, artificial lymphocytes, artificial immune system, affinity distance threshold, life counter transition function, genetic algorithms, classification system, negative selection, positive selection, mature, memory

Opsomming

Die hoofdoel van die biologiese immuun stelsel is om die liggaam te beskerm teen enige ongewenste vreemde selle wat die liggaam kan binnedring en sodoende skade aan die liggaam veroorsaak. Die immuun stelsel in die menslike liggaam het verskillende limfosiete wat die ongewenste selle raaksien en vernietig. Dit is moontlik om die biologiese immuun stelsel te modeleer as 'n kunsmatige immuun stelsel wat gebruik kan word om enige ongewenste patrone in 'n nie-biologiese omgewing raak te sien. Een van die hoof funksies van die biologiese immuun stelsel is om die struktuur van die ongewenste selle aan te leer om sodoende 'n vinniger immuun reaksie teenoor moontlike toekomstige ongewenste selle met naastenby of presies dieselfde struktuur te hê. Die kunsmatige immuun stelsel (KIS) kan dus gesien word as 'n patroonherkenningstelsel. Die KIS gebruik kunsmatige limfosiete (KLS) wat enige patroon kan klassifiseer as deel van 'n vooraf-bepaalde stel patrone al dan nie. In die immuun stelsel het die limfosiete verskillende toestande: Onvolwasse, Volwasse, Geheue en Vernietig. Die KLS'e wat in die vernietig-toestand is, moet van die aktiewe stel van KLS'e verwyder word. Die proses om van een toestand na 'n ander toestand oor te gaan moet op 'n doeltreffende wyse bepaal en beheer word. Die verhandeling lê 'n KIS voor om ongewenste patrone met 'n dinamiese aktiewe stel van KLS'e te herken en stel 'n toestands-veranderingsfunksie voor om die toestand van 'n KLS te bepaal. Die KIS in die verhandeling maak gebruik van evolusionêre komputasie tegnieke om 'n optimale stel van KLS'e te evoleer wat ongewenste patrone beter kan herken, en verwyder KLS'e in die vernietig-toestand vanuit die aktiewe stel van KLS'e.

Studieleier: Prof. A.P. Engelbrecht
Departement Rekenaarwetenskap
Graad: Magister Scientiae

Acknowledgments

- My Father in heaven for all my privileges and His mercy.
- My dad, mother and Dollie for their support.
- Prof. Andries Engelbrecht for his guidance and patience.
- CIRG Team, especially Werner van Rensburg and Nelis Franken.
- Telkom SA Ltd. - Centre of Excellence for their financial support.

"If all your peers understand what you've done, it's not creative."

- H. Heimlich

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification is the process to orderly separate a group of similar patterns into classes according to structure or characteristics common to each class. Various classification models have been developed, which can be broadly divided into two classes of algorithms. The first class of algorithms include decision trees [70], rough sets [58, 61] and rule induction [83], while the second class includes algorithms that model a natural process. Examples of these are the artificial neural network that models the biological neural network in the brain [6], evolutionary computation techniques that model the natural evolution of organisms [1] and swarm intelligence that models the behavior of a structured collection of interacting organisms to solve a global objective [47]. The modeling of natural processes has also proven to be successful in classification problems, optimisation problems, control, pattern matching and data mining. These computational techniques are usually trained with negative and positive examples that have been pre-classified according to a specific concept or rule. This training method is known as supervised learning and the trained model must be able to correctly predict or classify any pattern not seen before. Classification models have also been developed using unsupervised learning algorithms where the training process consists of automatically discovering similar patterns in data without relying on an external teacher [51].

Recently, artificial immune systems (AIS) have been developed as an alternative classification algorithm. An AIS is modeled after the natural immune system (NIS) to detect foreign patterns in a non-biological environment. The NIS has the ability to not only learn valid patterns and recognise foreign patterns (or anomalies), but also has the ability to memorise general foreign pattern structures [56, 63]. Contrary to standard classification algorithms, the AIS can be trained on positive patterns alone. After training on positive patterns, the artificial immune system can detect or distinguish negative patterns from the positive patterns. The AIS can thus be used as a classification algorithm where one class (the positive patterns) is separated from all other classes (the negative patterns).

The AIS has mostly been used in anomaly pattern detection where detectors are trained with negative selection using a set of positive patterns [16, 28, 29, 37, 39, 40, 49, 50, 74, 75]. The detectors are randomly initialised or constructed from a genetic library [38]. The patterns represented by the trained detectors can also be used as training data for other classification algorithms [16]. The detectors can also be trained with a training set of positive and negative patterns [38, 64].

The AIS proposed in this thesis uses a set of mature artificial lymphocytes (ALCs) to detect negative patterns. An ALC in the AIS becomes mature after the ALC has been trained on a set of positive patterns. This set must represent a good distribution of positive patterns or a complete representation of positive patterns. The ALCs can be trained with negative or positive selection. The ALCs with poor detection of negative patterns need to be distinguished from the ALCs with a good detection of negative patterns. It is important to distinguish between these types of ALCs to remove the ALCs with poor detection of negative examples from the set of ALCs, i.e. to guarantee an optimal set of ALCs with a high probability to detect negative patterns. This dissertation has as its main objective to develop an AIS where an optimal initial set of ALCs are evolved using a genetic algorithm (GA). The goal of the GA is to evolve individual ALCs with the least overlap with existing ALCs in the set and with the maximum space coverage of possible negative examples. The GA was chosen as optimisation method since the GA guarantees local optimum solutions. With each new ALC that must be added to the set of existing ALCs, the least overlap restriction forces the GA to explore different regions in the search space that are not yet covered by the existing set of ALCs. These evolved ALCs are then trained using positive or negative selection, and it is determined how well the trained set of ALCs perform on a number of classification problems. As a sub-objective, the dissertation proposes a method to dynamically determine the status of an ALC, which can be annihilated, mature or memory. The status of an ALC is determined based on the number of negative patterns detected, the space covered by the ALC and the number of patterns presented to the ALC to classify. The status of an ALC indicates the detection performance of the ALC. Annihilated ALCs are poor detectors and are removed from the ALC set.

The rest of the dissertation is organised as follows:

- *Chapter 2* introduces the natural immune system and explains how the natural immune system protects the body against viruses, bacteria and any pathogenic material that can damage the body. The different types of lymphocytes and the life cycle of a lymphocyte are discussed.

- *Chapter 3* gives an overview of evolutionary computation. The chapter presents the different recombination operators, selection methods and also gives a summary of the different evolutionary computation (EC) paradigms. Sufficient background on EC is provided only to support the development of the approach to evolve a set of ALCs.

- *Chapter 4* gives background on existing artificial immune system models or models that originated from ideas from immunology. Applications of the artificial immune system are also discussed.

- *Chapter 5* explains how artificial lymphocytes cover the non-self space and how their receptors are initialised. The training of an artificial lymphocyte is explained. The requirements for classifying a non-self pattern are also presented, and the hit ratio function is introduced. The three status types that artificial lymphocytes can assume in the artificial lymphocyte's life cycle are prioritised into low, medium and high. The life counter threshold function, that determines an artificial lymphocyte's state, is presented and explained.

- *Chapter 6* explains how one of the evolutionary computation paradigms (as discussed in chapter 3) will be used in the proposed genetic artificial immune system (GAIS) to evolve an initial set of ALCs.

- *Chapter 7* presents experimental results to illustrate and to discuss the behavior of GAIS on different classification problems which were collected from the UCI Machine Learning Repository [7]. The influence of the GAIS parameters on its performance is also investigated.

- *Chapter 8* concludes this dissertation and presents ideas relating to possible future work.

- *Appendix A* lists the publications derived from this dissertation.

- *Appendix B* lists and defines the symbols used throughout this dissertation.

- *Appendix C* lists and defines the abbreviations used throughout this dissertation.

- *Appendix D* lists and defines the main terms used throughout this dissertation.

Chapter 2

The Natural Immune System

"While lions pounce on zebras and robins peck at worms, the leukocytes in our own body devour invading germs"
- Perspective in Biology and Medicine 32:61, 1988.

The body has many defense mechanisms, among others are the skin of the body, the membrane that covers the hollow organs and vessels and the immune system. The immune system reacts to a specific foreign body material or pathogenic material (referred to as antigen). During these reactions a 'memory' is built up of regular encountered antigen. The obtained memory speeds up and improves the reaction of the immune system to future exposure to the same antigen. Due to this reason defense reactions are divided into three types: non-specific defense reactions, inherited defense reactions and specific defense reactions [56]. The immune system forms part of the specific defense reactions. The classical view of the immune system is that the immune system distinguishes between what is normal (*self*) and foreign (*non-self* or antigen) in the body. The recognition of antigens leads to the creation of specialised activated cells which inactivate or destroy these antigens. The natural immune system mostly consists of lymphocytes and lymphoid organs. These organs are the tonsils and adenoids, thymus, lymph nodes, spleen, Peyer's patches, appendix, lymphatic vessels and bone marrow. Lymphoid organs are responsible for the growth, development and deployment of the lymphocytes in the immune system. The lymphocytes are used to detect any antigens in the body. The immune system works on the principle of a pattern recognition system, recognising *non-self* patterns from the *self* patterns [63]. Recently Matzinger [54, 55] introduced the *danger theory*. The main idea of the *danger theory* is that the immune system distinguishes between what is dangerous and non-dangerous in the body. The *danger theory* differs from the classical view in that the immune system does not respond to all

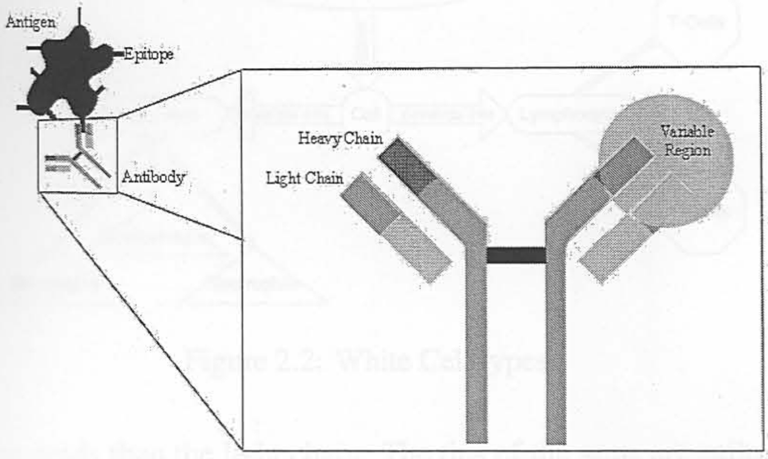


Figure 2.1: Antigen-Antibody-Complex

foreign cells, but only to those foreign cells that are harmful or dangerous to the body. The rest of this chapter explains the development of the different cell types in the immune system, antigens and antibodies, immune reactions and immunity types and the detection process of foreign body material.

2.1 Antibodies and Antigens

Within the natural immune system, antigens are material that can trigger immune response. An immune response is the body's reaction to antigens so that the antigens are eliminated to prevent damage to the body. Antigens can be either bacteria, fungi, parasites and/or viruses [71]. An antigen must be recognised as foreign (*non-self*). Every cell has a huge variety of antigens in its surface membrane. The foreign antigen is mostly present in the cell of micro-organisms and in the cell membrane of 'donor cells'. Donor cells are transplanted blood cells obtained through transplanted organs or blood. The small segments on the surface of an antigen are called *epitopes* (as shown in Figure 2.1). Epitopes trigger a specific immune response and antibodies bind to these epitopes [56].

Antibodies are chemical proteins. In contradiction to antigens, antibodies form part of *self* and are produced when lymphocytes come into contact with antigen (*non-self*). An antibody has a Y-shape (as shown Figure 2.1). Both arms of the Y consist of two identical heavy and two identical light chains. The chains are distinct into *heavy* and *light* since the heavy chain contains double

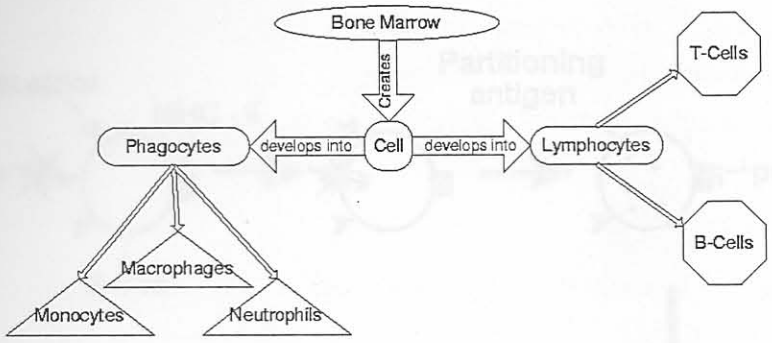


Figure 2.2: White Cell types

the number of amino-acids than the light chain. The tips of the arms are called the variable regions and vary from one antibody to another [71]. The variable regions enable the antibody to match antigen and bind to the epitopes of an antigen. After a binding between an antibody and an antigen's epitope, an antigen-antibody-complex is formed, which results into the de-activation of the antigen [56]. There are five classes of antibodies: IgM, IgG, IgA, IgE, IgD [56].

2.2 The White Cells

All cells in the body are created in the bone marrow (as illustrated in Figure 2.2). Some of these cells develop into large cell- and particle-devouring white cells known as phagocytes [71]. Phagocytes include monocytes, macrophages and neutrophils. Macrophages are versatile cells that secretes powerful chemicals and plays an important role in T-Cell activation. Other cells develop into small white cells known as lymphocytes.

2.2.1 The Lymphocytes

There are two types of lymphocytes: the T-Cell and B-Cell, both created in the bone marrow. On the surface of the T-Cells and B-Cells are receptor molecules that bind to other cells. The T-Cell binds only with molecules that are on the surface of other cells. The T-Cell first become mature in the thymus, whereas the B-Cell is already mature after creation in the bone marrow. A T-Cell becomes mature if and only if it does not have receptors that bind with molecules that represent *self* cells. It is therefore very important that the T-Cell can differentiate between *self* and *non-self* cells. Both T-Cells and B-Cells secrete lymphokines and macrophages secrete monokines. Monokines and lymphokines are known as cytokines and their function is to encourage cell growth, promote cell activation or destroy target cells [71]. These molecules on the surface of a

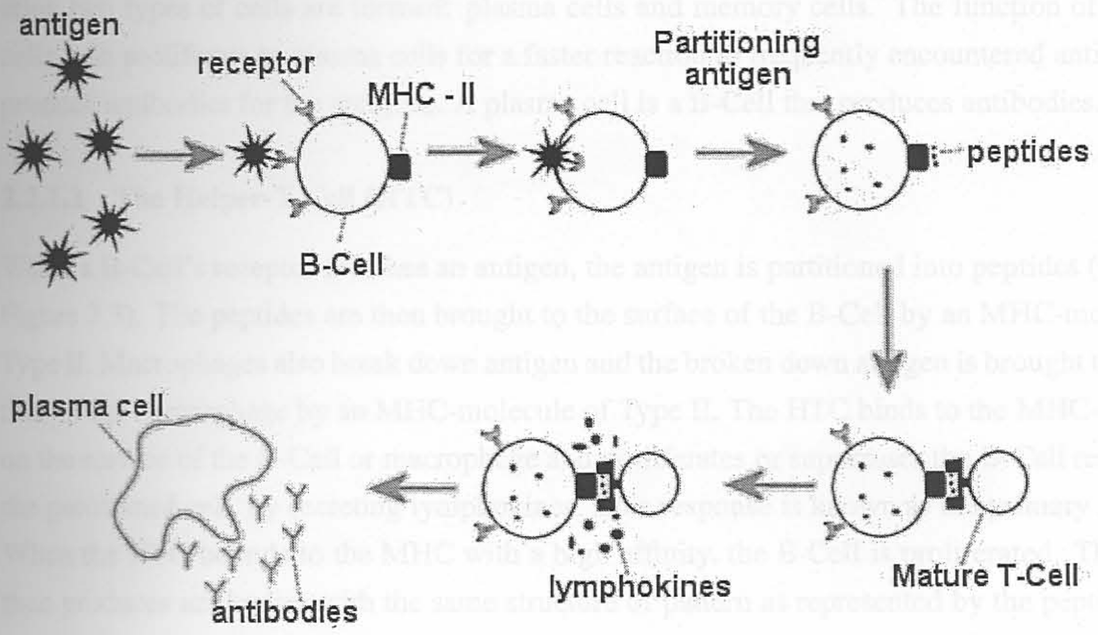


Figure 2.3: B-Cell develops into plasma cell, producing antibodies

cell are named the Major Histocompatibility Complex Molecules (MHC-molecules). Their main function is to bring to light the internal structure of a cell. MHC-molecules are grouped into two classes: Type I and Type II. MHC-molecules of Type I is on the surface of any cell and MHC-molecules of Type II mainly on the surface of B-Cells [63]. There are two types of T-Cells: The Helper-T-Cell and Natural-Killer-T-Cell. Each of these types of lymphocytes are described in detail below.

2.2.1.1 The B-Cell

The B-Cells are created in the bone marrow with monomeric IgM-receptors on their surfaces. A monomeric receptor is a chemical compound that can undergo a chemical reaction with other molecules to form larger molecules. In contrast to T-Cells, B-Cells leave the bone marrow as mature lymphocytes. B-Cells mostly exist in the milt and tonsils. It is in the milt and tonsils that the B-Cells develop into plasma cells after the B-Cells came into contact with antigens. After developing into plasma cells, the plasma cells produce antibodies which are effective against antigens [56]. The B-Cell has antigen-specific receptors and recognises in its natural state the antigens. When contact is made between B-Cell and antigen, clonal proliferation on the B-Cell takes place and is strengthened by Helper-T-Cells (as explained in section 2.2.1.2). During clonal proliferation two types of cells are formed: plasma cells and memory cells. The function of memory cells is to proliferate to plasma cells for a faster reaction to frequently encountered antigens and produce antibodies for the antigens. A plasma cell is a B-Cell that produces antibodies.

2.2.1.2 The Helper-T-Cell (HTC)

When a B-Cell's receptor matches an antigen, the antigen is partitioned into peptides (as shown Figure 2.3). The peptides are then brought to the surface of the B-Cell by an MHC-molecule of Type II. Macrophages also break down antigen and the broken down antigen is brought to the surface of the macrophage by an MHC-molecule of Type II. The HTC binds to the MHC-molecule on the surface of the B-Cell or macrophage and proliferates or suppresses the B-Cell response to the partitioned cell, by secreting lymphokines. This response is known as the primary response. When the HTC bounds to the MHC with a high affinity, the B-Cell is proliferated. The B-Cell then produces antibodies with the same structure or pattern as represented by the peptides. The production of antibodies is done after a *cloning process* of the B-Cell.

When the HTC does not bind with a high affinity, the B-Cell response is suppressed. Affinity is a force that causes the HTC to elect a MHC on the surface of the B-Cell with which the HTC has a stronger binding to unite, rather than with another MHC with a weaker binding. A higher affinity implies a stronger binding between the HTC and MHC. The antibodies then bind to the antigens' epitopes that have the same complementary structure or pattern. Epitopes are the portions on an antigen that is recognised by antibodies. When a B-Cell is proliferated enough, i.e. the B-Cell frequently detects antigens, it goes into a memory status, and when it is suppressed frequently it becomes annihilated and replaced by a newly created B-Cell. The immune system uses the B-Cells with memory status in a secondary response to frequently seen antigens of the same structure. The secondary response is much faster than the primary response, since no HTC signal or binding to the memory B-Cell is necessary for producing antibodies [63].

2.2.1.3 The Natural-Killer-T-Cell (NKTC)

The NKTC binds to MHC-molecules of type I (as illustrated in Figure 2.4). These MHC-molecules are found on all cells. Their function is to bring to light any viral proteins from a virally infected cell. The NKTC then binds to the MHC-molecule of Type I and destroys not only the virally infected cell but also the NKTC itself [63].

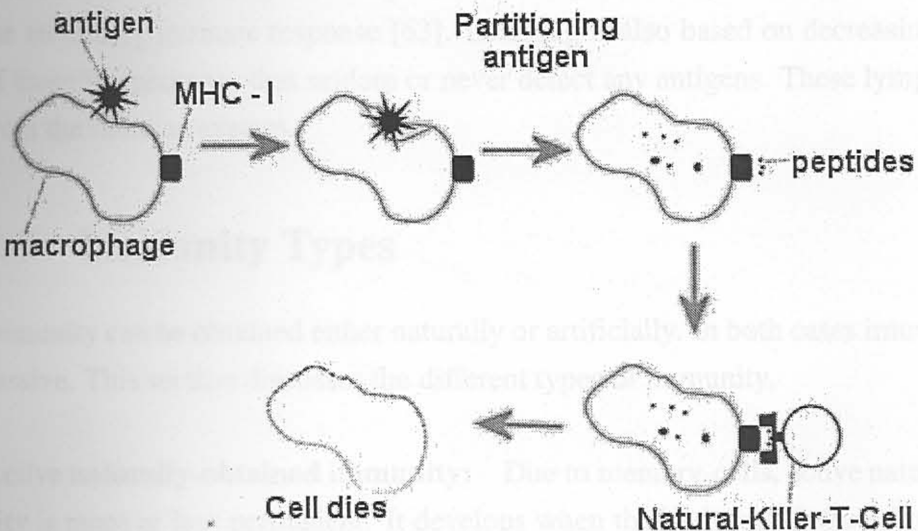


Figure 2.4: Macrophage and NKTC

2.3 The Cloning Process of the Lymphocyte

The cloning process is more generally known as *clonal selection*, which is the proliferation of the lymphocytes that recognise the antigens. The interaction of the lymphocyte with an antigen leads to an activation of the lymphocyte where upon the cell is proliferated and grown into a clone. Lymphocytes in a clone produce antibodies if it is a B-Cell and secrete growth factors (lymphokines) in the case of an HTC. Since antigens determine or select the lymphocytes that need to be cloned, the process is called *clonal selection* [56]. The fittest clones are those that bind to antigen best. For the process to be successful, the receptor molecule repository needs to be as complete and diverse as possible to recognise any foreign shape [63].

2.4 Learning the Antigen Structure

Learning in the immune system is based on increasing the population size of those lymphocytes that frequently recognise antigens. The immune system learns from experience the shape of the frequently encountered antigens and moves from a random receptor creation to a repertoire that represents the antigens more precisely. Since the total number of lymphocytes in the immune system is regulated, the increase in size of some clones decreases the size of other clones. This leads to the immune system forgetting previously learned antigens. When a familiar antigen is detected, the immune system responds with larger cloning sizes. This response is referred to as the secondary immune response [63]. Learning is also based on decreasing the population size of those lymphocytes that seldom or never detect any antigens. These lymphocytes are removed from the immune system.

2.5 Immunity Types

Immunity can be obtained either naturally or artificially. In both cases immunity can be active or passive. This section discusses the different types of immunity.

Active naturally-obtained immunity: Due to memory-cells, active naturally-obtained immunity is more or less permanent. It develops when the body gets infected or receives foreign red blood cells and actively produces antibodies to deactivate the antigen [56].

Passive naturally-obtained immunity: Passive naturally-obtained immunity is short-lived since antibodies are continuously broken down without creation of new antibodies. New antibodies are not created because the antigens did not activate the *self* immune system. The immunity type develops from IgG-antibodies that are transplanted from the mother to the baby. The secreted IgA-antibodies in mothers-milk are another example of this immunity type and protect the baby from any antigens with which the mother came into contact [56].

Active artificially-obtained immunity: Active artificially-obtained immunity develops when dead organisms or weakened organisms are therapeutically applied. The concept is that special treated organisms keep their antigens without provoking illness-reactions [56].

Passive artificially-obtained immunity: Passive artificially-obtained immunity is obtained when a specific antibody which was produced by another human or animal, is injected into the body for an emergency treatment. Immunity is short-lived, since the immune system is not activated [56].

2.6 Conclusion

This chapter explained the working of the natural immune system and how the immune system protects the body against viruses, bacteria and any pathogenic material that can damage the body.

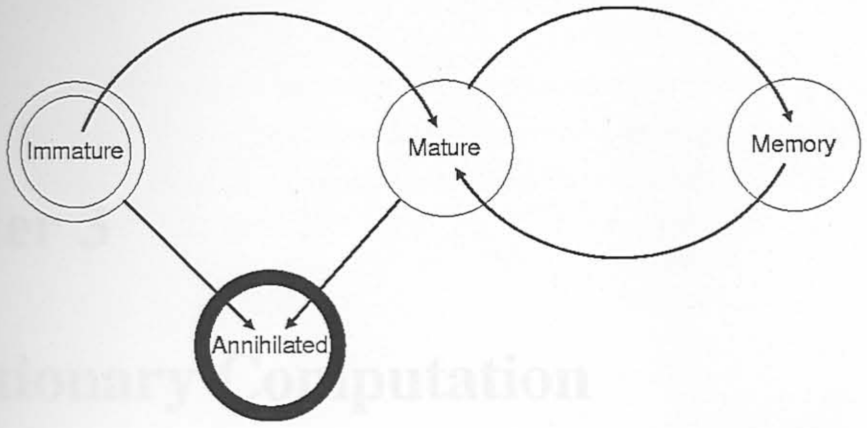


Figure 2.5: Life cycle of a lymphocyte

The different types of lymphocytes and molecules in the immune system were discussed. From this discussion it can be summarised that lymphocytes have different states: Immature, Mature, Memory and Annihilated (Figure 2.5 illustrates the life cycle of lymphocytes). The next chapter gives an overview of evolutionary computation (EC). An evolutionary algorithm (EA) is used in the Genetic Artificial Immune System (GAIS) developed in this dissertation (chapter 6) to evolve lymphocytes that can match non-self patterns with a higher affinity in a non-biological environment.

Chapter 3

Evolutionary Computation

"survival of the fittest" -
The Origin of Species
by Charles Darwin, 2001

This chapter gives an overview of evolutionary computation (EC) [2]. The different recombination operators, selection methods and mutation are explained. The chapter also summarises the different EC paradigms. Section 3.1 provides a summary of a general evolutionary algorithm (EA), while succeeding sections provide a more detailed discussion of aspects of evolutionary computation. In these sections the focus will be on genetic algorithms, since this EC paradigm is used in the GAIS classifier to evolve a set of ALCs.

Evolutionary computation mimics natural evolution in biological organisms. Evolutionary algorithms (EA) are stochastic search algorithms where the search for an optimal solution is guided by the principle of survival of the *fittest* and mathematical models of genetic and behavioral inheritance. A population of candidate solutions (or individuals) is evolved for a number of *generations* through application of a number of operators such as *crossover*, *mutation* and *selection* until an optimal, or best individual is found as solution to the optimisation problem. The four most common specific paradigms of EC are [72]: Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Programming (EP) and Evolutionary Strategies (ES). These paradigms use the same general evolutionary algorithm.

3.1 A General Evolutionary Algorithm

The following pseudo-code algorithm summarises a general evolutionary algorithm. Although all operator types are included in the algorithm, different EC paradigms use different operators and different representations of the chromosome. The differences between the most popular EC paradigms are discussed in section 3.2.

General EA:

```
1. set the generation counter, $g=0$
2. initialise a population of $K$ chromosomes
3. while no convergence
    (a) evaluate the fitness of each chromosome in the population
    (b) perform crossover
        i. select parents
        ii. produce offspring from the selected parents
    (c) perform mutation
        i. select a candidate chromosome
        ii. mutate selected candidate
    (d) select the survivors to form the next generation
    (e) evolve the next generation ($g=g+1$)
```

Several aspects of the general algorithm above need to be explained in more detail, for example the initialisation of the population, the fitness of a chromosome, the crossover operator and mutation on chromosomes, the selection of survivors for the next generation and the convergence in the population. These aspects are discussed in sections 3.3 to 3.8. First, the differences between EC paradigms are discussed in section 3.2.

3.2 EC Paradigms

This section outlines the main differences between the most popular EC paradigms. Other paradigms include differential evolution [66], cultural evolution [42] and co-evolution [25].

3.2.1 Genetic Algorithms (GA)

Genetic algorithms were first introduced by Alex Fraser [32], while Holland laid down the basic principles in 1975 [41]. The genetic algorithm mimics genetic evolution [4]. Chromosomes are in genotype-space and are mostly bit-strings. Continuous-valued variables are usually coded into a binary representation. A drawback of bit-string chromosomes is the occurrence of Hamming cliffs [26], which is addressed by using Hamming bit-string representations. The genetic algorithm uses recombination operators, mutation and selection methods to evolve to the best solution [26, 72].

3.2.2 Genetic Programming (GP)

Genetic programming also models genetic evolution but the chromosomes represent executable programs in a tree structure. Genetic programming was developed by Koza [52]. The goal of the genetic programming algorithm is to evolve the best executable program in a problem domain that performs (or executes) best. Therefore the fitness function measures how well a chromosome executes in the problem domain. Crossover is implemented by swapping subtrees between selected parents to produce offspring. Mutation can be implemented as randomly changing a node, expanding (growing) the tree to a larger depth, truncating the tree or randomly replace a terminal node [26, 72].

3.2.3 Evolutionary Programming (EP)

Different from genetic algorithms, evolutionary programming models behavioral evolution and chromosomes represent solutions in phenotype-space. EP was developed by Fogel [27]. EP does not use crossover, since each individual in the population generates one offspring through the mutation operator. The offspring compete against the chromosomes of the previous generation for survival to the next generation. The EP uses *elitism* as replacement scheme [26, 72].

3.2.4 Evolutionary Strategies (ES)

Evolutionary strategies model the evolution of evolution [67, 73]. That is, ES is concerned with optimising the evolutionary process itself [26, 68]. Chromosomes present solutions in both genotype- and phenotype-space, but the fitness of the chromosomes is the behavior of the individual in phenotype-space. The chromosome consists of genetic material and strategy parameters. The ES algorithm evolves the chromosome's strategy parameters, and the strategy parameters is used to evolve the genetic material of the chromosome. The ES algorithm also uses mutation as operator and a mutated chromosome is only accepted if mutation has improved the fitness of the individual [26, 72]. The ES uses different crossover, mutation and selection techniques [26].

3.3 The Chromosome

An evolutionary algorithm makes use of a population of chromosomes, or individuals. Each chromosome represents a potential solution to the problem that needs to be optimised. A chromosome consists of the set of parameters to the problem or function that needs to be optimised. The parameters are known as *genes* and represent values in the same space as the function being optimised (referred to as *phenotype-space*). Each gene consists of *alleles*. Alleles are specific values from the domain of the corresponding parameter assigned to the gene.

There are different chromosome representation schemes. Some of these are binary string representations where the binary values are discretised real numbers or boolean values, trees that represent programs, or the chromosome can represent real-valued variables. The chromosomes converge in the limit and the best chromosome is chosen as a solution or an approximate solution to the problem. Parameter-values are usually mapped to an intermediate space, referred to as *genotype-space*. This mapping from phenotype-space to genotype-space is known as *coding* and the inverse mapping is known as *decoding* [72].

When the problem that needs to be optimised is a mathematical function, then the chromosome will represent the real-valued variables of that function. For a more complex problem like the optimisation of the execution of a program, the chromosomes will represent different programs using a tree-structure. The coding mappings can influence the global behavior of the EA algorithm. It is important to include all necessary parameters in the chromosome representation to prevent evolution to a less optimal solution. Operations on the chromosome produces offspring to widen the search space of solutions. The chromosome's genotypes are evaluated on the corresponding phenotype to determine the chromosome's fitness. At each generation, the chromosome's fitness is calculated and only the fittest chromosomes survive to the next generation. The next section discusses the fitness function, followed by methods of offspring creation.

3.4 Calculating the Fitness

The fitness function maps the chromosome's representation into a scalar value. The scalar value of the chromosome indicates how close a chromosome is to the optimal solution. The fitness function therefore provides a quantification of the quality of the chromosome. It is the fitness of a chromosome that determines whether the chromosome will be selected to produce offspring and quantifies its chances for survival among the other chromosomes in the population to the next generation. The probability to mutate a chromosome is usually a function of the chromosome's fitness. Chromosomes with high scalar fitness values should preferably not be mutated.

The fitness function is problem-specific. The function can either be a unimodal function or a multi-modal function. A unimodal function has a single optimal solution where a multi-modal function has multiple optima. The standard optimisation methods only find a single solution and for local optimisation algorithms this solution is either a local minimum or local maximum, thus not necessarily the best solution [9]. Global optimisation algorithms find the best solution. A technique known as *niching* has been developed to find multiple solutions in multi-modal functions. GAs have been successfully applied with *niching*, with the effect that individuals converge to different solutions, or *niches* [43, 53]. The best individual per *niche* is one of the solutions to the problem. Niching can be done in two ways: Parallel and sequential niching. Parallel niching concurrently finds niches in the search space through strategies that identify and refine potentially good solutions over time [35, 43, 53, 57]. Sequential niching develops niches over time, in sequence. After each discovered niche, individuals are repelled from the area around the niche to focus on unexplored areas in the search space [3].

Some problems have several objectives which are represented as sub-objectives in a multi-objective function. In most cases the sub-objective functions are in conflict, i.e. the reduction of one objective results in the increase of another objective. Multi-objective optimisation techniques find solutions with a good trade-off between the conflicting sub-objectives. Multi-modal functions and multi-objective problems have the same goal of finding multiple solutions for the optimisation problem. Multi-objective optimisation techniques find multiple solutions for a number of sub-objectives as niching has the objective to find multiple solutions for a single objective for a multi-modal problem. It is of utmost importance to include all necessary objectives in the fitness function, to prevent evolving a less optimal solution for the problem. For more information on multi-objective optimisation, the reader is referred to [13].

Before the fitness of a chromosome can be calculated, the chromosome's representation first needs to be decoded to phenotype-space if the domain of the fitness function is in phenotype-space. If the fitness function is too complex to evaluate, an approximate function evaluation can be used, which approximately gives the same value as the "true" fitness function in less time [12]. Penalty functions have also been considered in the evaluation of fitness, penalising a chromosome for an invalid or "bad" solution to the problem or a chromosome that violates a certain set of restrictions [34, 69].

3.5 Reproduction

Reproduction is the process of producing new offspring from selected individuals through crossover or mutation. The reproduction step consists of a selection step where parents are selected and a crossover step where genetic material of individuals are exchanged to form offspring. These operators are discussed in more detail in this section.

3.5.1 Selection

This section describes the most popular selection methods that can be used to select chromosomes as parents for crossover, individual chromosomes for mutation or the chromosomes that survive to the next generation. Usually, chromosomes with high fitness are selected for crossover to converge faster to a best solution. Highly fit chromosomes should not be selected for mutation to prevent the danger of diverging from "good" solutions in the search space. Therefore chromosomes with low fitness are usually selected for mutation.

All selection methods are based on the fitness of the chromosomes [10]. Certain selection methods allow selection of a chromosome more than once, resulting in clones of the selected chromosome. An advantage of clones is that more chromosomes with high fitness form part of the population, while a disadvantage is the probability of less diversity in the search space. When selecting chromosomes to survive to the next generation, either all offspring are selected to replace some or all parents from the previous generation, or chromosomes from both the offspring and the parents from the previous generation are considered for selection. Offspring can only replace parents if the offspring has a higher fitness than the parents. Goldberg and Deb [36] compared many of these selection methods. The most popular selection techniques are listed below:

Random Selection: All the individuals in the population have an equal chance to be selected with no reference to their individual fitness.

Proportional Selection: The probability of an individual to be selected is based on the individual's fitness proportional to the summed fitness of all the individuals in the population. A drawback of this selection method is that an individual may dominate the production of offspring which results in a limited diversity among individuals in the new population. This drawback can be overcome by limiting the number of offspring the selected individual may produce.

Rank-Based Selection: The individuals in the population are sorted according to their fitness. The rank ordering of the individuals in the sorted set determines the probability to select an individual. Selection is thus not based on the magnitude of an individual's fitness. As such, ranking has the advantage that a highly fit individual will not dominate the selection process. An alternative approach is to assign survival probabilities to the individuals in the sorted set using an exponential function with the rank as parameter to the exponential function. This results in a higher selection intensity with the disadvantage of less diversity which could lead to suboptimal solutions.

Tournament Selection: A random sample of i individuals is selected from the population to take part in a tournament of selecting the individual with the best fitness. The sample may be taken with or without replacement. It is therefore possible that an individual can combine with itself to produce offspring or a parent can be selected more than once to produce offspring. An advantage of tournament selection is that the individuals with worst fitness will not be selected and as long as sampling is done with replacement, the best individual will not dominate the reproduction process.

Elitism: This selection method is used to select a set of the best individuals from the previous population that will survive to the next generation. The number of individuals in the selected set is known as the *generation gap*, ς. These chromosomes are not mutated. If $\varsigma = 0$, then the new generation consists entirely out of new individuals. If $\varsigma > 0$, then ς individuals survive to the next generation. These ς selected individuals are either the ς best individuals to ensure that the maximum fitness in the population does not decrease or ς individuals selected with one of the selection methods explained above.

3.5.2 Crossover

The crossover operator exchanges genetic material between two or more selected parents to produce offspring. The main idea of the crossover operator is to recombine genetic material between fit chromosomes from previous generations with a certain probability, p_c, to produce even fitter offspring. The crossover process is described by the following general algorithm:

1. generate $\xi \sim U(0,1)$
2. select parents A and B
3. if $\xi > p_c$ then crossover is not performed and A and B are returned, otherwise goto step 4
4. exchange genetic material between A and B according to one of the crossover operators (described below)
5. return the produced offspring

The following section explains how the crossover operator is applied to chromosomes with different representations.

3.5.2.1 Continuous-valued Chromosomes

Arithmetic crossover can be used if genes are continuous-valued [5]. A number of arithmetic operators have been developed, for example:

- Average - take the arithmetic average of the genes in parents A and B as the new value for the gene in the offspring O. That is, $O_i = \frac{A_i+B_i}{2.0}$, where i is the index to the i-th gene in the chromosome.
- Geometric mean - take the square-root of the product of the two gene-values in parents A and B as the new value for the gene in the offspring O. That is, $O_i = \sqrt{A_i * B_i}$, where i is the index to the i-th gene in the chromosome.
- Extension - take the difference between the two gene-values in parents A and B, add the difference to the higher of A or B, or subtract the difference from the lower of A or B. That is,
$$O_i = Max(A_i, B_i) + y$$

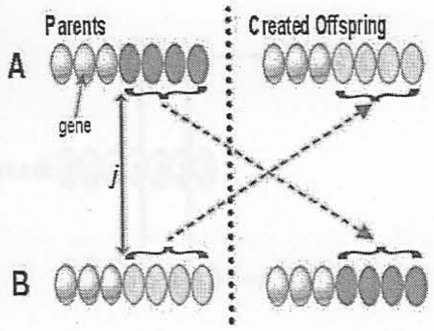


Figure 3.1: One-point crossover

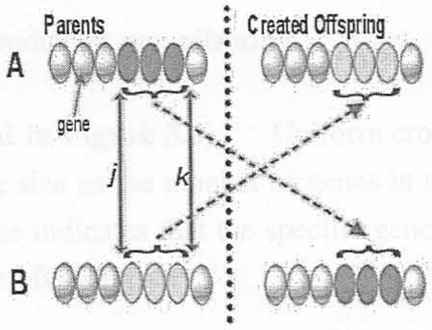


Figure 3.2: Two-point crossover

or

$$O_i = Min(A_i, B_i) - y$$

where $y = |A_i - B_i|$.

3.5.2.2 Nominal-valued Chromosomes

Nominal-valued chromosomes have genes which assume values from a finite set of values. Binary-valued genes is a special case, where the set of values contains only two values. Popular crossover operators include [26]:

One-point crossover (illustrated in Figure 3.1): A random position, j, is selected in both parents A and B. The tail of parent A is swapped with the tail of parent B, producing two offspring.

Two-point crossover (illustrated in Figure 3.2): Two random positions, j and k, are selected in both parents A and B. The selected segment between j and k in parent A is swapped with the

117341796
b1633677x

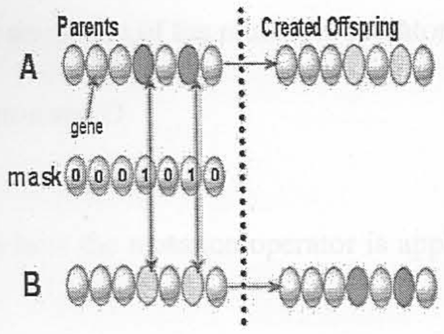


Figure 3.3: Uniform crossover

selected segment in parent B, producing two offspring.

Uniform crossover (illustrated in Figure 3.3): Uniform crossover uses a randomly generated bit mask that has the same size as the number of genes in the chromosomes. The gene in the mask that has a value of one indicates that the specific genes have to be swapped between parents A and B, producing two offspring.

Crossover prevents the evolutionary process to randomly search for the optimal solution. A high p_c can result in premature convergence to suboptimal solutions. If p_c is too low, the evolutionary process may tend to have low diversity in the search space. This may restrict the possibility to find an optimal solution in the search space.

3.6 Mutation

The main objective of mutation is to introduce diversity into a population of chromosomes with a certain probability p_m. The mutation operator randomly changes the genetic representation of the selected chromosome to ensure diversity and to cover larger parts of the search space. Mutation is only used in certain EC paradigms, where it is considered as a background operator. In these EC paradigms (excluding EP), mutation operates on offspring that have been produced by the crossover operator. The following algorithm summarises mutation:

```
1. select a chromosome $D$ to mutate
2. for each gene in $D$ generate $\xi \sim U(0,1)$
    (a) if $\xi > p_m$ then do not mutate the gene, otherwise goto 2.(b)
```

(b) Mutate the gene by using one of the mutation operators (described below)

3. return the mutated chromosome D

The following section explains how the mutation operator is applied to chromosomes with different representations.

3.6.1 Continuous-valued Chromosomes

Arithmetic mutation can be used if the genes of the chromosome are continuous-valued [5]. Arithmetic mutation has been implemented in the following ways:

- Random replacement, which replaces the value of a gene with a new random value that is valid in the domain of the variable.
- Creep mutation, where a small random value is added or subtracted from the value of a gene.
- Geometric creep mutation, where the value of a gene is multiplied with a value close to 1.

The random values can be sampled from any of a number of probability distributions, e.g. uniform (within a specified range), exponential, Gaussian, normal, binomial, etc. Usually, a Gaussian distribution is used with a zero mean. To obtain a small value a small variance is used. The variance of the distribution is usually a function of the fitness of the individual that needs to be mutated. An individual with a high fitness will be mutated less than an individual with a lower fitness, which will be mutated more.

3.6.2 Nominal-valued Chromosomes

Different mutation operators for nominal-valued genes are [26]:

Random Mutate (illustrated in Figure 3.4): Random genes in the chromosome are selected and each gene's value is replaced with its complement or a new random value (with probability p_m).

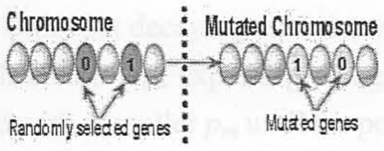


Figure 3.4: Random Mutation

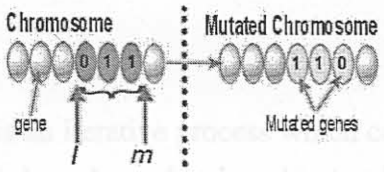


Figure 3.5: In-order Mutation

In-order Mutate (illustrated in Figure 3.5): Two random positions, *l* and *m*, are selected in the chromosome and only genes between these positions are considered for replacement using random mutation.

A high p_m causes large diversity in the search space, since new genetic material is more rapidly introduced. A large mutation rate may cause good genetic material to be lost. A high mutation rate is however beneficial in situations where more rapid coverage of the search space is needed. Usually, one starts with a large initial mutation rate, decreasing it over time. A high mutation rate basically results in a random search.

3.7 The Initial Population

The initial population is generated by randomly selecting valid values for the genes from the domain of the corresponding variables. Random selection ensures that the initial population of chromosomes has a good uniform coverage of the search space. Domain knowledge of the search space can be used to initialise the values of the genes, using heuristics to bias the initial population to potentially good solutions. The drawback of using domain knowledge is that the potential good areas in the search space are missed. A better solution may exist in the missed areas. The size of the initial population influences the convergence speed of the algorithm. With a large population size, the search space is well covered. The larger diversity may result in less generations with longer time per generation to converge to the best solution. The consequence of a small population size is less diversity in the search space. The small population may take more generations to converge but with less time per generation. To explore a larger search space with a small population size, an initial large, but decaying mutation rate can be used. The individuals will then initially be mutated with a large p_m, exploring a larger search space. As generations pass by the individuals are mutated with a smaller p_m until the population converges to an optimal solution.

3.8 Convergence

An evolutionary algorithm (EA) is an iterative process which continues until a convergence criterion is satisfied. Several methods have been developed to test if an EA should terminate. Some of the most frequently used criteria are summarised below:

- The EA terminates when a maximum number of generations is exceeded. When the maximum number of generations is too small, the EA might not have evolved an optimal solution yet, i.e. no convergence in the population. When the maximum number of generations is too high, the population in the EA might have already converged to an optimal solution before reaching the maximum number of generations, thus wasting computational effort.

- The desired, or an acceptable best chromosome evolves. When the optimum solution to a problem is known, it can be used as a measure to determine when a suitable or satisfying solution has evolved.

- The average fitness of the population and the variance of the population's fitness do not change over a certain number of generations. This indicates that the fitness of the population has stabilised.

3.9 Conclusion

This chapter gave an overview on evolutionary computation (EC). The chapter presented the different recombination operators, selection methods and also gave a summary of the different EC paradigms. EC methods require only the values of the function that needs to be optimised. This fact makes EC methods stochastic, which can potentially find the global optimum of the objective function. EC methods search in parallel for a global solution to the problem. A disadvantage of EC methods is the computational cost, since a large number of function evaluations must be performed to find a satisfying result. The choice of evolutionary operators, chromosome representation and fitness function have a critical impact on the performance of EC methods.

Since EC methods are stochastic in nature there is no guarantee of convergence to the optimum solution. An alternative method to solve complex problems is the artificial immune system.

The next chapter gives background on existing artificial immune system models and their application to non-biological environments. An artificial immune system is a dynamic distributed system that is capable to learn and recognise patterns. The AIS is robust, scalable and tolerant to noise.

Chapter 4

The Artificial Immune System

The artificial immune system (AIS) is a computational system which is applied to problem domains. The AIS mimics the models of the natural immune system's immune functions and principles. In chapter 2 of this dissertation, the natural immune system (NIS) was discussed. The NIS is a very complex system that is capable of learning the structure of normal cells (self patterns) and classifying foreign cells (non-self patterns). The NIS also builds up a 'memory' of frequently seen non-self patterns to ensure a faster secondary immune response to non-self patterns with identical or similar structure. Although the NIS is not yet fully understood, research has shown that the NIS consists of mature T-Cells and B-Cells that co-operate to detect any foreign cell in the body (as explained in section 2.2.1). The B-Cells produce antibodies through a process known as clonal selection (as discussed in section 2.3).

The capabilities of the NIS to distinguish between normal cells and foreign cells (with only having knowledge on what is normal) and learning the non-self cells' structure, inspired the modeling of the NIS into an AIS for application in non-biological environments. A major advantage of the AIS is that the model only needs to be trained on positive examples (knowledge on the self patterns) to detect or classify non-self patterns in a non-biological environment. A drawback is that a limited knowledge of positive examples or a bad representation of positive examples can lead to misclassification of non-self patterns.

This chapter provides some background information on the different existing artificial immune system models, as well as models inspired by immunology and applications of the artificial immune system.

4.1 Natural to Artificial

There are a few theories surrounding the structure and functioning of the NIS due to its biological complexity. The NIS has many different lymphocytes that form part of the immune response to a foreign cell. Two of these lymphocytes - the T-Cell and B-Cell - have been clearly defined and their functioning in the NIS identified (as explained in section 2.2.1). In the NIS it is mainly the inner working and co-operation between the mature T-Cells and B-Cells that is responsible for the secretion of antibodies as an immune response to antigens. The T-Cell becomes mature in the thymus. A mature T-Cell is self-tolerant, i.e. the T-Cell does not bind to self cells. The mature T-Cell's ability to discriminate between self cells and non-self cells makes the NIS capable of detecting non-self cells. When a receptor of the B-Cell binds to an antigen, the antigen is partitioned and then brought to the surface with an MHC-molecule. The receptor of the T-Cell binds with a certain affinity to the MHC-molecule on the surface of the B-Cell. The affinity can be seen as a measurement to the number of lymphokines that must be secreted by the T-Cell to clonally proliferate the B-Cell into a plasma cell that can produce antibodies. The memory of the NIS on frequently detected antigen is built-up by the B-Cells that frequently proliferates into plasma cells. Thus, to model the proposed artificial immune system, there are a few basic concepts that must be considered:

- There are trained detectors (artificial lymphocytes) that detect non-self patterns with a certain affinity.
- The artificial immune system needs a good repository of self patterns or self and non-self patterns to train the artificial lymphocytes (ALCs) to be self-tolerant.
- The affinity between an ALC and a pattern needs to be measured. The measured affinity can indicate to what degree an ALC detects a pattern.
- To be able to measure affinity, the representation of the patterns and the ALCs need to have the same structure.
- The artificial immune system has memory that is built-up by the artificial lymphocytes that frequently detect non-self patterns.
- When an ALC detects non-self patterns, it can be *cloned* and the clones can be mutated to have more diversity in the search space.

In [29, 39, 40] similar architectures for an AIS are proposed and used for security.

The measured degree of affinity between an ALC and a pattern needs to exceed a certain degree for the pattern to be detected by an ALC. There are different approaches to measure affinity and represent patterns in the AIS. Timmis *et al.* [79] represented patterns as vectors of floating points. The Euclidean distance between an ALC and a pattern was used as a measure of affinity. De castro and Von Zuben [21] used binary strings to represent patterns. The affinity between an ALC and a pattern was measured using the hamming distance. The above mentioned measurements indicate the similarity between an ALC and the pattern. A lower value calculated by the above measurements indicates a stronger affinity between an ALC and the pattern. Hunt and Cooke [44] used the same representation of binary strings, but like Forrest *et al.* [30] the measurement of affinity was done with the r-continuous matching rule. The r-continuous matching rule is a partial matching rule: An ALC detects a pattern if there are r-continuous or more matches in the corresponding positions. r is the degree of affinity for an ALC to detect a pattern. A higher value of r indicates a stronger affinity between an ALC and the pattern.

The proposed AIS has a set of ALCs that is used to detect non-self patterns. The ALC set is trained with a training set that consists of self patterns, non-self patterns or self and non-self patterns. Each pattern in the training set has been labeled as self or non-self. The ALCs can be trained to become self-tolerant like a mature T-Cell or to detect non-self patterns with a higher affinity. The trained ALC set is then used to detect non-self patterns. Hightower *et al.* [38] and Oprea [60] used a training set that consisted of non-self patterns. A genetic algorithm was used to evolve ALC sets with a larger detection ratio of non-self patterns in the training set. Forrest *et al.* [30] used a technique known as negative selection to train ALCs to become self-tolerant. The training set consisted of self patterns represented by nominal valued attributes or binary strings. The ALCs are randomly generated and tested against the training set of self patterns. If the ALC does not detect any of the self patterns in the training set, it is added to the ALC set. The training set is monitored by continually testing the ALC set against the training set for changes. A negative selection method to train an ALC with continuously-valued self patterns are presented by Gonzalez *et al.* [37]. The continuously-valued negative selection method evolves ALCs that are the furthest away from the training set of self patterns and that are separated to maximise the non-self space coverage. A randomly generated ALC that is trained with negative selection does represent a pattern in non-self space, but not necessarily an antigen. A different approach is proposed by Kim [49] where ALCs are not randomly generated and tested with negative selection, but an evolutionary process is used to evolve ALCs towards non-self and to maintain diversity and generality among the ALCs. The model by Potter and De Jong [64] applies a co-evolutionary genetic algorithm to evolve ALCs towards the selected class of non-self patterns in the training set and further away from the selected class of self patterns. Once the fitness of the ALC set evolves to a point where all the non-self patterns and none of the self patterns are detected, the ALCs represent a description of the concept. If the training set of self and non-self patterns is noisy, the ALC set will be evolved until most of the non-self patterns are detected and as few as possible self patterns are detected. The evolved ALCs can discriminate between examples and counter-examples of a given concept. Each class of patterns in the training set is selected in turn as self and all other classes as non-self to evolve the different concept in the training set. The negative selection method used in the proposed AIS, evolves ALCs that are binary-valued to cover the maximum non-self space with the furthest distance from the training set of self patterns and the least overlap among the evolved ALCs.

The clonal selection of the immune system was modeled by an algorithm proposed by de Castro [21, 24]. The presented algorithm, CLONALG, performs machine-learning and pattern recognition tasks. The training set consists of non-self patterns. The ALCs in the ALC set are randomly initialised and the ALCs that recognised the selected non-self pattern in the training set with the highest affinity, are cloned. The clones are mutated and then a new population of ALCs are selected from the mutated clones and the previous ALCs, according to their affinity with the selected non-self pattern in the training set. A selection of ALCs with high affinity go into the memory pool. New randomly initialised ALCs are inserted into the set of ALCs for the next non-self training pattern to be recognised. The algorithm can also be used to solve complex problems, eg. multi-modal function optimisation.

Timmis introduced the concept of artificial recognition balls (ARBs) in a resource limited artificial immune system [78]. An ARB has the same representation as an ALC, but stands for a number of identical ALCs. Thus, each ARB allocates a number of resources based on its stimulation level. The total number of resources of the system is bounded. Watkins adopted the concept of ARB's [81]. To be able to identify a memory cell, a training set of antigens (non-self patterns) is presented to the system. A subset of the training set is used to create an initial batch of memory cells. Each of the remaining non-self training patterns are then matched against the memory cells. The memory cell with the closest match to a specific non-self training pattern is then cloned to form an ARB. The level of cloning is determined by the strength of the affinity between the non-self pattern and the memory cell. The newly created ARB is then added to a pool of existing ARBs of the same class. Resources are allocated to an ARB depending on the affinity between the ARB and the presented non-self pattern as well as the class of the non-self pattern. The ARBs are clonally expanded until the average stimulation level of the ARBs is above a certain threshold. When the limit for available resources has been reached by the ARBs, resources are removed from the ARBs with the lowest affinity until the limit is no longer exceeded. The ARBs with bad performance will have no resources allocated to them and are removed from the pool. The ARBs in the pool are evaluated on their affinity with a non-self pattern. If the best matching ARBs in the pool have a higher affinity with the presented non-self pattern than the best matching memory cell in the memory pool, then the ARBs are added to the memory pool. Only the memory pool is used to classify non-self patterns in the test set.

All of the above models are supervised training methods except the work by de Castro and the work by Timmis, which are unsupervised. CLONALG and AINE were both algorithms for clustering. The training sets consist either of self patterns, non-self patterns or both. The following section introduces the network theory of interconnected B-Cells in the natural immune system.

4.2 The Network Theory

The theory of Jerne is that the B-Cells are interconnected to form a network of cells [45, 62]. When a B-Cell in the network respond to a foreign cell, the activated B-Cell stimulates all the other B-Cells to which it is connected in the network. Work that has been done in AISs on the network theory of B-Cells can be found in [77, 79, 80]. Timmis [77] implemented the network theory with interconnected ARBs. When two ARBs have a high affinity between them, a link is established between them. Therefor, a network of ARBs is formed based on the similarity and affinity among the ARBs. In the network of ARBs, closely related ARBs form clusters that represent clusters in the data set. This model resulted in a successful unsupervised training method to visualise data. Another unsupervised approach to cluster data was done by de Castro and Von Zuben [19]. In this model, named aiNet, the B-Cells formed part of an edge-weighted graph. Some of the B-Cells were connected with edges, with a weight (connection strength) assigned to each edge. Thus, the graph is not necessarily fully connected. The network is trained by representing non-self patterns to the interconnected B-Cells. The distance between B-Cells with the highest affinity for the non-self pattern are then decreased. The immune network theory has also been successfully developed and applied to optimise multi-modal functions [17].

4.3 The Genetic Artificial Immune System

The algorithm proposed in this dissertation, named GAIS - Genetic Artificial Immune System, represents all patterns in space as binary vectors and uses the hamming distance as affinity measurement. A GA is used to evolve ALCs with maximum non-self space coverage and minimum overlap among existing ALCs (as explained in chapter 6). The ALCs are trained with an adopted negative selection method (as explained in section 5.2.2.1) or with positive selection (as explained in section 5.2.2.2). The affinity threshold of an ALC is used to determine a match with a non-self pattern. With the adopted negative selection method the affinity threshold is determined by the distance to the closest self pattern from the ALC. The algorithm is supervised with a training set that consists of self patterns. GAIS is different from existing AIS models in that the GA does not evolve ALCs towards non-self patterns (as in the model of Kim [49]), neither does it evolve ALCs with a larger detection ratio of non-self patterns in the training set (as in the models of Hightower *et al.* [38] and Oprea [60]). The GA in GAIS evolves ALCs with a maximum non-self space coverage and a minimum overlap with existing ALCs. The main goal is thus to evolve mature ALCs to detect non-self patterns that have not been presented to GAIS during training. Surely all evolved ALCs will cover non-self space, but not all ALCs will detect non-self patterns. Therefor, a proposed transition function, the life counter function (as explained in section 5.3), determine an ALC's status (as defined in section 5.2.1). ALCs with annihilated status are removed in an attempt only to have mature and memory ALCs with optimum classification of non-self patterns. GAIS has the advantage to classify patterns in a problem space where only positive patterns are available for training.

4.4 Applications and Other Models

The artificial immune system has been successfully applied to many problem domains. Some of these domains range from network intrusion and anomaly detection [16, 28, 29, 37, 39, 40, 49, 50, 74, 75] to data classification models [65, 82], (the model of Pramanik *et al.* [65] bridges the models of [59] and [48]), virus detection [30], concept learning [64], data clustering [19], robotics [46, 80], pattern recognition and data mining [11, 44, 77, 79]. The AIS has also been applied to the initialisation of feed-forward neural network weights [23], the initialisation of centers of a radial basis function neural network [22] and the optimisation of multi-modal functions [17, 33]. The interested reader is referred to [15, 18, 20] for more information on AIS applications.

4.5 Conclusion

The chapter gave a background overview on the existing theories of the functioning of the natural immune system and the modeling thereof, as well as the successful applications of the AIS in non-biological environments. The next chapter will explain the modeling of the T-Cell and B-Cell as artificial lymphocytes in the AIS. The different states in the life cycle of an artificial lymphocyte is discussed. The next chapter also presents a threshold function that is used to determine the state of an artificial lymphocyte.

Chapter 5

The Artificial Lymphocyte Life Counter

"As a first approximation, I define "belief" not as the object of believing but as the subject's investment in a proposition, the act of saying it and considering it as true."

- Thomas Carlyle

This chapter explains how artificial lymphocytes (ALCs) cover the non-self space, how their receptors are initialised and how their *affinity distance thresholds* are trained. The discrimination between *self* and *non-self* space is defined. The life cycle of an ALC is discussed and the life counter threshold function is introduced. The life counter threshold function determines in which state an ALC is in its life cycle. The requirements and the training of an ALC to classify a pattern as *non-self*, are also explained.

5.1 Introduction

The natural immune system has mature or memory lymphocytes with receptors on each lymphocyte's surface that function as pattern detectors or classifiers. These receptors distinguish between *self* (normal cells) and *non-self* (antigens) patterns by only binding to *non-self* patterns, i.e. the mature or memory lymphocyte does not bind to any *self* (normal cell). The receptors bind to the *non-self* pattern with a certain affinity. A lymphocyte binding with a higher affinity than another indicates that the lymphocyte has a better detection of the *non-self* pattern. The task of the artificial lymphocyte (ALC) is to be able to classify any pattern as *self* or *non-self* by detecting only *non-self* patterns and leave known self patterns undetected. In the context of computational pattern recognition, *self*, S, represents all patterns in a finite space, $\mho$, that is legitimate in a non-biological environment and *non-self*, $\overline{S}$, represents all patterns that is not in *self*.

That is,

$$S \cup \overline{S} = \mho$$

For an ALC to detect a non-self pattern, the receptor of the ALC must match the non-self pattern with a certain affinity.

The purpose of the AIS is to create an optimal set of ALCs that can detect and classify *non-self* from *self* (without detecting *self*) and to remove ALCs that do not classify any patterns or have a low probability to detect any *non-self* patterns. The AIS creates ALCs by randomly initialising the receptors and sets the *affinity distance threshold* by using a known *self* set of patterns to guarantee that created ALCs do not detect any of the self patterns in the known *self* set. Usually the set of ALCs is static. The ALCs are randomly initialised and their receptors are trained with a training set that consists of *non-self* and *self* patterns using negative selection or positive selection, i.e. supervised learning. The negative selection method selects patterns that do not match *self*. The static set of trained ALCs is then tested to determine classification performance. A larger set of trained ALCs results in better classification and a too small set of trained ALCs result in worse classification. The unsupervised approach to train the ALCs with positive or negative selection also uses a static set of randomly initialised ALCs during training. A larger set of randomly initialised ALCs classifies the training set better than a smaller set of randomly initialised ALCs. The clusters formed by the ALCs are then tested with a test set. It is assumed in this dissertation that the receptor of an ALC is a binary vector and that all patterns that need to be classified are in binary format. It is also assumed that the AIS has a static incomplete *self* set for training. That is, incremental learning is not considered.

The rest of this chapter describes the architecture of the artificial lymphocyte (ALC). The sections to follow introduce the different states in the ALC's life cycle and discusses the different selection techniques to train an ALC. A threshold function, the *Life Counter* (*LC*), is also presented and the influence of the parameters on the function is explained. The threshold function determines the status of an artificial lymphocyte within the lymphocyte life cycle. The AIS uses the status of an ALC to determine if the ALC must be removed from the set of ALCs.

5.2 The Artificial Lymphocyte

As stated in section 5.1, the purpose of an ALC is to detect and classify non-self patterns from self patterns. Since the B-Cell has receptors on its surface to bind to antigen and the T-Cell has receptors on its surface to match viral proteins that are represented by MHC-molecules as peptides, both B-Cell and T-Cell lymphocytes are modeled into a general artificial lymphocyte with a bit-string receptor. The receptor has a length k, equal to the length of a self and non-self pattern. The receptor has an affinity-distance threshold (ADT) that must be met to detect non-self patterns. In other words, the affinity of an ALC to a pattern must be greater than or equal to ADT for a match to occur, in which case the pattern is considered as non-self. Section 5.2.3 explains the role of the ADT in detecting non-self patterns. The artificial lymphocyte's receptor cannot bind or match a non-self pattern with an affinity that is greater than the size of the pattern, therefore $ADT \leq k$. The receptor of an ALC is randomly initialised to represent any pattern in the search space. That is, before an ALC can detect non-self patterns, it needs to be trained not to detect self patterns. Training is done by measuring the receptor of an ALC against the static incomplete self set to determine the ALC's affinity-distance threshold, as discussed in section 5.2.2. This guarantees that the ALC will not overlap with any of the known self patterns, and that the ALC's initial status is mature. The ALC's status is determined by a threshold function called the *Life Counter* (*LC*), which is explained in section 5.3. The status of an ALC indicates whether the ALC must be replaced with a newly constructed or evolved ALC, or be given higher priority than other ALCs.

5.2.1 Life Cycle of the Artificial Lymphocyte

This section introduces and explains the different states an ALC can have during its life cycle. The status of an ALC, which can be memory, mature or annihilated, determines its priority in a group of ALCs. The status of a trained ALC can be one of three stages:

Annihilated (Low priority): An ALC with this status has either been mutated to detect a self pattern or it has not detected any patterns as non-self and is then declared as obsolete in the group of ALCs. Annihilated ALCs must be removed from the group of ALCs.

Mature (Medium priority): The mature-ALC detects non-self patterns on a regular basis. An ALC in the mature status will be demoted to annihilated status if the ALC starts to detect no non-self patterns. If the ALC starts to detect non-self patterns more frequently it will be promoted to memory status again.

Memory (High priority): Memory-ALCs detect non-self patterns more frequently than mature-ALCs. Memory-ALCs are given higher priority than other ALCs by evaluating an incoming pattern first before other ALCs, resulting in a faster non-self detection. These ALCs are not mutated nor replaced by any newly constructed ALC. An ALC in memory status cannot be annihilated and first needs to be demoted to mature status. This happens when an ALC in memory status does not frequently detect non-self patterns as before, resulting in demotion to mature status.

An ALC with immature status has not yet been trained by the AIS and can therefore not be used to detect any non-self patterns. ALCs with annihilated status (low priority) have a higher probability to be removed from the set of ALCs than ALCs with mature status (medium priority). As long as an ALC has memory status (high priority), the ALC will never be removed from the set of ALCs, since the memory ALC has a higher probability in classifying a pattern than an ALC with mature status or even annihilated status. Therefore it is important to determine, at any time, which ALCs are in memory status. The LC threshold function is used to determine the status of an ALC, and to translate an ALC from one state to another (as is discussed in section 5.3).

5.2.2 Training the ALC to Cover Non-Self Space

An ALC can be trained with one of two selection methods: negative selection or positive selection. Both negative and positive selection methods determine the best ADT for the ALC. A non-self pattern can be incorrectly classified by an ALC as a self pattern. This misclassification is known as a *false negative*. A self pattern can also be incorrectly classified by an ALC as a non-self pattern. This misclassification is known as a *false positive*. For purposes of training, an incomplete static self set is used and the bit-string receptor is randomly initialised. The two selection methods are described next.

5.2.2.1 Adapted Negative Selection

The adapted negative selection method (refer to Figure 5.1) trains an ALC to have a maximum affinity-distance threshold, a_{neg}, that does not overlap with the self set. All patterns with a hamming distance from the ALC that is less than a_{neg}, are detected as non-self. To guarantee a maximum a_{neg} with no overlap with self, a_{neg} is set to the hamming distance of the nearest self

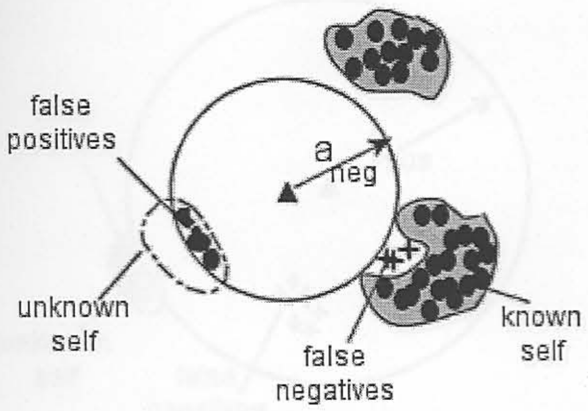


Figure 5.1: Venn-diagram of Adapted Negative Selection ALC

pattern to the ALC. Figure 5.1 shows that, with the adapted negative selection, the self pattern with the smallest hamming distance to the ALC is used to determine the maximum a_{neg}.

5.2.2.2 Positive Selection

The positive selection method (refer to Figure 5.2) trains an ALC to have a minimum affinity-distance threshold, a_{pos}, that does overlap with the self set. All patterns with a hamming distance from the ALC that is greater than a_{pos}, are detected as non-self. To guarantee a minimum a_{pos} with overlap with self, a_{pos} is set to the hamming distance of the self pattern furthest from the ALC. Figure 5.2 shows that, with positive selection, the self pattern with the biggest hamming distance to the ALC is used to determine the minimum a_{pos}.

Figures 5.1 and 5.2 illustrate the drawback of *false positives* and *false negatives* when the respective selection methods train the ALCs. These drawbacks are due to an incomplete static self set. The *known self* is the incomplete static self set that is used to train the ALCs and the *unknown self* is the self patterns that are not known during training. The *unknown self* can also represent self patterns which are outliers to the set of *known self* patterns.

5.2.3 Matching a Non-Self Pattern

In the natural immune system, a B-Cell that regularly detects antigen goes into a memory status. This memory B-Cell is used in a secondary response to detect antigen faster than the primary response. It is therefore important to keep record of the number of non-self pattern matches (or

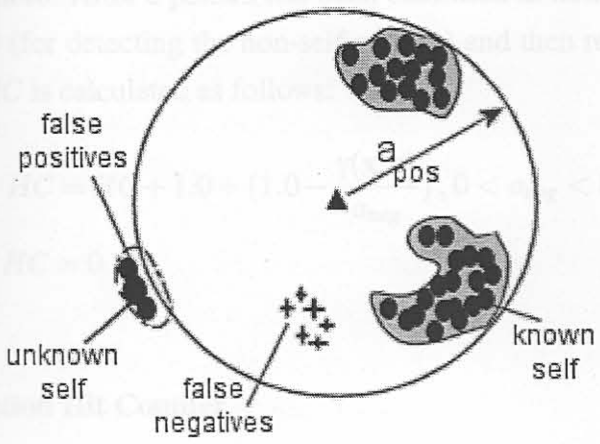


Figure 5.2: Venn-diagram of Positive Selection ALC

detections) made by an ALC to determine the ALC's matching ratio after classifying a number of non-self patterns. The *HitCounter* (*HC*) of an ALC is updated to keep record of the number of matches. The hamming distance between a pattern and the ALC receptor can be used to determine if the ALC detects a non-self pattern, defined as follows:

$$\gamma(\mathbf{x},\mathbf{r}) = \sum_{i=1}^{i\leq k} XOR(x_i, r_i)$$

where **x** is the bit-string of the pattern that needs to be classified, **r** is the bit-string of the receptor, *XOR* is the exclusive-or between the bits of the two bit-strings, *i* is the bit-index and *k* is the size of a pattern. Section 5.2.3.1 and Section 5.2.3.2 describe the requirements for an ALC to detect a non-self pattern and how the *HitCounter* of the ALC is updated for negative and positive selection respectively.

5.2.3.1 Adapted Negative Selection Hit Counter

As discussed in section 5.2.2.1, for an ALC trained using the adapted negative selection to detect a non-self pattern, the following relation must hold: $\gamma(\mathbf{x},\mathbf{r}) < a_{neg}$, i.e. the non-self pattern needs to be closer to the ALC than the closest self pattern to that ALC. A smaller value of $\gamma(\mathbf{x},\mathbf{r})$ indicates a more exact match to the pattern of the receptor. Then, $\frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$ is the difference ratio of a pattern with the receptor. The complement of $\frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$, which is $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$, is the similarity ratio between a pattern and the receptor. The *HC* of an ALC that matches a non-self pattern more exact (i.e. $\gamma(\mathbf{x},\mathbf{r}) \to 0$, $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}} \to 1.0$) than an ALC where $\gamma(\mathbf{x},\mathbf{r}) \to a_{neg}$, must be incremented (rewarded) more. After a pattern has been classified as non-self, the *HC* of the ALC is incremented with 1.0 (for detecting the non-self pattern) and then rewarded by the similarity ratio $1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}$. The *HC* is calculated as follows:

$$HC = HC + 1.0 + (1.0 - \frac{\gamma(\mathbf{x},\mathbf{r})}{a_{neg}}), 0 < a_{neg} < k$$

with the initial value of $HC = 0$.

5.2.3.2 Positive Selection Hit Counter

As discussed in section 5.2.2.2, for an ALC trained using positive selection to detect a non-self pattern, the following relation must hold: $\gamma(\mathbf{x},\mathbf{r}) > a_{pos} \Rightarrow \gamma(\mathbf{x},\mathbf{r}) - a_{pos} > 0.0$, i.e. the non-self pattern needs to be further away from the ALC than the furthest self pattern to that ALC. A larger value of $\gamma(\mathbf{x},\mathbf{r})$ indicates a less exact match to the pattern of the receptor. Since a_{pos} indicates the maximum ADT to a self pattern, the complement $k - a_{pos}$, indicates the maximum ADT to a non-self pattern. Then, $\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}$ is the difference ratio of a pattern to the receptor. The *HC* of an ALC that matches a non-self pattern less exact (i.e. $\gamma(\mathbf{x},\mathbf{r}) \rightarrow k$) than an ALC where $\gamma(\mathbf{x},\mathbf{r}) \rightarrow a_{pos}$, is incremented (rewarded) more. After a pattern has been classified as non-self, the *HC* of an ALC is incremented with 1.0 (for detecting the non-self pattern) and then rewarded by the difference ratio $\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}$. The *HC* is calculated as follows for positive selection:

$$HC = HC + 1.0 + (\frac{\gamma(\mathbf{x},\mathbf{r}) - a_{pos}}{k - a_{pos}}), 0 < a_{pos} < k$$

with the initial value of $HC = 0$.

5.2.3.3 The Hit Ratio

The *HitRatio* (*HR*) calculates the ratio at which an ALC matched non-self patterns. The hit ratio of an ALC is calculated after a specified number of patterns has been classified, the *IterationSize* (*IS*). A *HitRatio* (*HR*) is then calculated with parameters *HC* and *IS* as follows:

$$HR(HC, IS) = \frac{HC}{IS}$$

The HR is calculated to determine an ALC's detection ratio of non-self patterns in an iteration, i.e. the number of non-self patterns that were detected or matched by an ALC during an iteration.

5.3 The Life Counter

As stated earlier, the life counter (LC) determines in which state an ALC is at any given time. There are three states in which an ALC can be, prioritised into low, medium and high (as described in section 5.2.1). The LC is used to map the states to a continuous range $(0,1)$ (i.e. $m(LC) : \{low, medium, high\} \rightarrow (0,1)$). A life counter value closer to 1.0 indicates that the ALC has a higher priority than an ALC with a life counter value close to 0.0. The LC value of an ALC is calculated at specified time steps. The time step can be set to a constant iteration size (IS) of incoming patterns. The value of the LC depends on the ALC's HR, the *minimum matching ratio* (coverage of non-self space) of an ALC, and the IS.

For example, consider an ALC A with $a = 10$ and ALC B with $a = 5$ (assume that both A and B are initialised with the adapted negative selection). Let $k = 15$. A covers more non-self space than B since $a_A > a_B$. Therefore, it can be expected that A must have a higher HR than B. Suppose B has a higher HR than A, then B has a higher probability to gain memory status than A. For A to have the same status as B, $HR_A = 2 * HR_B$ since $a_A = 2 * a_B$. Therefore, an ALC converges to memory status if the number of classified non-self patterns is greater than the number of patterns in the non-self space covered by the ALC. An ALC converges to annihilated status if the number of classified non-self patterns is less than the number of patterns in the non-self space covered by the ALC.

The minimum matching ratio of an ALC to detect a non-self pattern also influences the status of the ALC. The minimum matching ratio, β, is calculated as follows:

- For an ALC trained with the adapted negative selection,

$$\beta_{neg} = 1.0 - \frac{a_{neg}}{k}$$

- For an ALC trained with positive selection,

$$\beta_{pos} = \frac{a_{pos}}{k}$$

After each IS patterns, the ALC's life counter is re-calculated to determine the ALC's state using the life counter threshold function (as explained below).

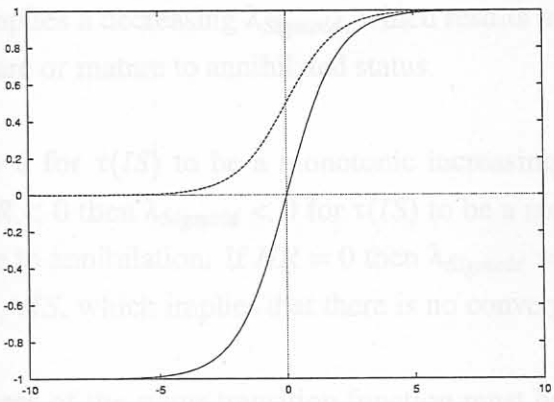


Figure 5.3: Hyperbolic Tangent Function and Sigmoid Function

5.3.1 Life Counter Threshold Function

The LC threshold function, τ, determines the status of an ALC based on the HR, β and IS of the ALC. The τ-value of the ALC is calculated after a number of patterns, IS, has been classified. The requirement for τ is that $\tau \in (0,1)$, thus a function with range of $(0,1)$ is needed, i.e.

$$\tau(IS) \rightarrow (0,1)$$

$\tau(IS)$ must be a continuous function and monotonic increasing from mature to memory status, and monotonic decreasing from memory to mature or mature to annihilation. The sigmoid function (dashed line in Figure 5.3),

$$g(\lambda_{Sigmoid}, x_{Sigmoid}) = \frac{1}{1+e^{(-\lambda_{Sigmoid} \times x_{Sigmoid})}},$$

where $\lambda_{Sigmoid}$ controls the steepness of $g(\lambda_{Sigmoid}, x_{Sigmoid})$, satisfies these requirements. For $g(\lambda_{Sigmoid}, x_{Sigmoid})$ to be a monotonic increasing function, $\lambda_{Sigmoid} > 0$ and for $g(\lambda_{Sigmoid}, x_{Sigmoid})$ to be a monotonic decreasing function, $\lambda_{Sigmoid} < 0$. Thus,

$$\tau(IS) = g(\lambda_{Sigmoid}, IS)$$

The steepness of $g(\lambda_{Sigmoid}, IS)$ represents the rate at which an ALC must converge to memory or annihilated status. As explained in section 5.2.3.3, the HR of an ALC calculates the ratio at which an ALC matched non-self patterns. Thus, the HR of an ALC determines the convergence ratio at which an ALC must converge to memory or annihilated status. An increasing HR implies an increasing $\lambda_{Sigmoid}$, which results in faster convergence of an ALC from mature to memory status. A decreasing HR implies a decreasing $\lambda_{Sigmoid}$, which results in faster convergence of an ALC from memory to mature or mature to annihilated status.

If $HR > 0$ then $\lambda_{Sigmoid} > 0$ for $\tau(IS)$ to be a monotonic increasing function, implying convergence to memory. If $HR < 0$ then $\lambda_{Sigmoid} < 0$ for $\tau(IS)$ to be a monotonic decreasing function, implying convergence to annihilation. If $HR = 0$ then $\lambda_{Sigmoid} = 0$ for $\tau(IS)$ to be a linear function, with $\tau(IS) = 0.5\,, \forall IS$, which implies that there is no convergence to memory or annihilation.

To achieve this, the steepness of the status transition function must be a function of HR with a range of $(-1, 1)$. The rate at which an ALC's status moves from one state to the next is influenced by the minimum matching ratio β. For increasing β the rate of a state transition should be faster than for a smaller β. A faster transition rate is obtained with a steeper gradient of the state transition function, which is achieved with $\lambda_{Sigmoid} > 0$. On the other hand, a slower transition (for smaller β) is obtained by using a $\lambda_{Sigmoid} < 0$. To achieve these effects, $\lambda_{Sigmoid}$ can be a function of the hyperbolic tangent function. The hyperbolic tangent function (solid line in Figure 5.3), is defined as

$$f(\lambda_{Hyper}, x_{Hyper}) = \frac{2}{1+e^{(-\lambda_{Hyper} \times x_{Hyper})}} - 1$$

where λ_{Hyper} controls the steepness of $f(\lambda_{Hyper}, x_{Hyper})$. Thus, $\lambda_{Sigmoid}$ can be defined as

$$\begin{aligned}\lambda_{Sigmoid} &= f(\beta, HR) \\ &= \frac{2}{1+e^{(-\beta \times HR)}} - 1\end{aligned}$$

In the above, β controls the steepness of the hyperbolic tangent function. A large value of β causes a higher rate of change of $f(\beta, HR)$, which ensures an increasing slope for the state transition function g. A smaller value of β causes a lower rate of change of $f(\beta, HR)$, which ensures a decreasing slope for the state transition function. Values of $HR > 0$ will cause a positive output for $f(\beta, HR)$, which ensures a monotonic increasing state transition function. On the other hand $HR < 0$ will cause a negative output for $f(\beta, HR)$, which ensures a monotonic decreasing state transition function.

In summary:

$$\begin{aligned}\tau(IS) &\doteq g(\lambda_{Sigmoid}, IS) \\ &= g(f(\beta, HR), IS)\end{aligned}$$

$$= \frac{1}{1+e^{(-f(\beta,HR)\times IS)}}$$

where $f(\beta,HR) = \frac{2}{1+e^{(-\beta\times HR)}} - 1$

The HR defined in section 5.2.3.3 will always have a positive value, since $IS > 0$ and $HC \geq 0$. With $HC \geq 0$ and $\beta \geq 0$, $f(\beta,HR) \geq 0$. This results in the fact that g will always be a monotonic increasing function. This means that the function g will always converge to 1.0 and all ALCs will either stay in the mature status or transolve to the memory status. To prevent all the ALCs to converge to memory, a constant detection ratio can be subtracted from HR. The constant detection ratio is called the expected matching ratio (EMR) and is explained in the following section.

5.4 The Expected Matching Ratio

The expected matching ratio (EMR) is the detection ratio of non-self patterns expected from an ALC. The EMR needs to be updated so that after each iteration, the EMR represents a more correct rate of non-self patterns that can be expected to be detected by an ALC. If η_h is the number of non-self patterns detected in the current iteration, then $\frac{\eta_h}{IS}$ calculates the matching rate of non-self patterns per iteration. A proposed function to update the EMR for the next iteration is to take the average over the current matching ratio and the current iteration's EMR, i.e.

$$EMR_{h+1} = \frac{EMR_h + \frac{\eta_h}{IS}}{2.0}$$

where EMR_{h+1} is the calculated expected matching ratio for the next iteration. The HR function is redefined to be also dependent on the EMR:

$$HR(HC,IS,EMR) = \frac{HC}{IS} - EMR$$

From the above equation it can be concluded that if an ALC detects less patterns as expected ($EMR > \frac{HC}{IS}$) then $HR < 0$, resulting in a monotonic decreasing g. If $EMR < \frac{HC}{IS}$ then $HR > 0$, which results in a monotonic increasing g. With the redefined HR, not all the ALCs will converge to memory, and those that match less non-self patterns as expected, move toward annihilated status.

5.5 Conclusion

This chapter explained how ALCs cover non-self space and how their receptors are initialised. The training of an ALC, using the adapted negative selection or positive selection, was explained. The requirements for classifying a non-self pattern were also presented in terms of the hit ratio function. The three types of status were prioritised into low, medium and high. Then the life counter threshold function was presented and explained. In the next chapter, the use of evolutionary computation techniques to evolve ALCs are discussed.

Chapter 6

Evolving Artificial Lymphocytes

This chapter proposes an approach to evolve an optimal initial set of ALCs using a genetic algorithm (GA). An optimal set of ALCs can be defined as a set of trained ALCs with the largest non-self space coverage and with minimum overlap among the ALCs in the set. Thus, the problem that needs to be optimised is multi-objective. GAIS (Genetic Artificial Immune System) uses a GA to search the problem space for solutions to these objectives. Figure 6.1 (page 50) illustrates the flow layout of the GAIS algorithm and the function of the GA within GAIS. GAIS has an initial empty set of ALCs. The set is populated by evolved ALCs which are obtained from the GA one-by-one (sequentially), until the set of ALCs has converged. The converged ALC set is used to classify any pattern as non-self or self. Therefor the GAIS algorithm has two phases:

- The evolutionary process to evolve the best ALC using a GA.
- The detection process of non-self patterns.

The first phase forms part of the training process of the ALCs. The role of the GA in GAIS is explained in section 6.1. After training, the evolved ALC set is used to classify any pattern as self or non-self. The classification process is discussed in section 6.2.

6.1 Genetic Algorithm to Evolve an ALC

The success and efficiency of GAIS is mainly influenced by the quality of the ALCs used to detect non-self patterns. An ALC is randomly initialised and trained with either the adapted negative selection method or positive selection method. The trained ALC usually forms part of a static set of trained ALCs which is used to detect non-self patterns. This section shows how a GA can be used to evolve a dynamic set of ALCs. The main objective of the GA is to evolve an

ALC to be added to the existing active ALC set such that the evolved ALC maximises its ADT (if the ALC was trained with the adapted negative selection method) or minimises its ADT (if the ALC was trained with the positive selection method) and to minimise the average overlap with existing ALCs in the active ALC set. In the case of the adapted negative selection, an ALC with the maximum hamming distance from the set of self patterns implies that the maximum non-self space is covered by the ALC without overlapping with the self set. In the case of positive selection, an ALC with the minimum hamming distance from the set of self patterns implies that the similarity between the evolved ALC's receptor and the set of self patterns needs to be maximised for maximum non-self space coverage. Additionally, the GA also maximises the distance between the new ALC and the ALCs in the active ALC set. Maximising the distance between the new ALC and the active ALC set guarantees that the evolved ALC has the lowest average overlap with the existing set of ALCs, and this forces greater coverage of non-self space. Thus, a GA is used in GAIS to optimise a multi-objective goal. The purpose of the GA is to evolve one optimal ALC to be added to the active set of ALCs.

The GA has an initial population of I ALCs. The initial population is randomly initialised (as shown in figure 6.1). The fitness of each ALC in the population is calculated to be proportional to the ADT and overlap with other ALCs (refer to section 6.1.3). The evolutionary process stops as soon as the maximum number of generations is reached or the fitness of the ALC population has converged. The fittest ALC in the population is then selected to be added to the active ALC set.

The following is the pseudo-code summary for the GA in phase 1 to evolve an ALC:

```
1. set $g=0$ ($g$ counts the number of generations)
2. randomly initialise $I$ chromosomes as population $H_g$
3. while no convergence in $H_g$
    (a) evaluate the fitness of each chromosome in $H_g$ using equation (6.1) or (6.2)
    (b) apply cross-over
        i. select two parents
        ii. produce offspring from the two selected parents and add them to the offspring
            set
    (c) apply mutation
```

   i. select a candidate chromosome from the offspring set

   ii. mutate the selected candidate

(d) select the new generation from the previous generation and the offspring set

(e) $g = g + 1$

The operators and other design aspects of the algorithm are explained in more detail below.

6.1.1 ALC as Chromosome

As explained in section 3.3, each chromosome represents a potential solution and consists of the set of parameters (genes) to the problem that needs to be optimised. *Alleles* are specific values from the domain of the corresponding parameter assigned to the gene.

As explained in Section 5.2, each ALC has a receptor that is a bit-string of fixed length k. These bit-strings are used to detect or match patterns as explained in section 5.2.3. Therefor all patterns that need to be classified by an ALC must first be coded to a bit-string. A technique known as *binning* is used to discretise patterns with floating-point values. *Binning* calculates the valid interval of values for each attribute c in the data set. The interval is calculated by subtracting the minimum value of the attribute from the maximum value of the attribute. The calculated interval of each attribute is then divided into b bins. Each bin represents a group of values. To determine into which group a value of the specific attribute falls, the following calculation is used

$$G(x_{c,j}) = \frac{x_{c,j} - min_{c=1,\ldots C}\{x_{c,j}\}}{max_{c=1,\ldots C}\{x_{c,j}\} - min_{c=1,\ldots C}\{x_{c,j}\}} * b$$

where $x_{c,j}$ is the floating-point value x of attribute c in pattern j and C is the number of attributes in a pattern.

The floating-point value in the pattern is now in nominal form. To code patterns from nominal to binary, the number of groups or bins an attribute can have is used to determine the number of bits necessary to represent an attribute's values. The number of bits is calculated as $N = \frac{log(b)}{log2}$ bits. Then the nominal value in each attribute can be coded to binary using standard binary encoding. The binary encoded chromosome in effect represents the receptor of an ALC.

6.1.2 The Initial Population

The receptor of an ALC is randomly initialised. The initial population of the GA is a set of these randomly initialised ALCs. Random selection ensures that the initial population of chromosomes has a good uniform coverage of the search space. The size of the population, I, is static throughout the evolutionary process.

6.1.3 Evaluating the Chromosome's Fitness

The ALCs in the AIS can be trained with one of two selection methods (as explained in section 5.2.2). There are two objectives that need to be optimised by the GA. The first objective to be optimised is to evolve an ALC with the largest average hamming distance, $\chi(D,\mathbf{r})$, from an existing set of ALCs. This objective ensures that the GA evolves an ALC with the lowest average overlap with an existing set of ALCs. The average hamming distance from an existing set of ALCs is calculated as follows

$$\chi(D,\mathbf{r}) = \frac{\sum_{l=1}^{l\leq P}\gamma(\mathbf{d}_l,\mathbf{r})}{P}$$

where D is the active ALC set, P is the number of ALCs in the active set D, $\mathbf{d}_l$ is the receptor of the l-th ALC in the active set and $\mathbf{r}$ is the receptor of the ALC from which the average hamming distance must be calculated. γ is the hamming distance between $\mathbf{d}_l$ and $\mathbf{r}$, defined as

$$\gamma(\mathbf{d}_l,\mathbf{r}) = \sum_{i=1}^{i\leq k} XOR(d_{l_i}, r_i)$$

where XOR is the exclusive-or between the bits of $\mathbf{d}_l$ and $\mathbf{r}$, i is the bit-index and k is the size of the receptor.

The second objective that needs to be optimised by the GA is to evolve an ALC with an optimised affinity distance threshold. The second objective differs for each selection method as explained below.

6.1.3.1 Adapted Negative Selection Fitness

In the case of the adapted negative selection, the GA needs to evolve an ALC that has the maximum affinity distance threshold a_{neg}. This objective ensures that the GA evolves an ALC with the maximum hamming distance from the set of self patterns. An ALC with maximum hamming distance from the set of self patterns implies that a maximum non-self space is covered by the ALC without overlapping with the self set.

The fitness function υ_{neg} for the adapted negative selection is then defined as

$$\upsilon_{neg} = w_1 a_{neg} + w_2 \chi(D, \mathbf{r}) \tag{6.1}$$

with $w_1 + w_2 = 1.0$ where $w_1, w_2 \in [0, 1]$. w_1 and w_2 are weights that determine the influence that a_{neg} and $\chi(D, \mathbf{r})$ have on the fitness of an ALC, respectively. Since there is no conflict between a_{neg} and $\chi(D, \mathbf{r})$, the sub-objectives are only weighted with w_1 and w_2. With $w_1 = 1.0$, $\chi(D, \mathbf{r})$ has no influence on the ALC's fitness and the fitness is determined only by a_{neg}. With $w_2 = 1.0$, a_{neg} has no influence on the ALC's fitness and the fitness is determined by $\chi(D, \mathbf{r})$. The value of w_2 is calculated as $w_2 = 1.0 - w_1$, and the value of w_1 is problem dependent, obtained through cross-validation.

6.1.3.2 Positive Selection Fitness

In the case of positive selection, the GA needs to evolve an ALC that has the minimum affinity distance threshold a_{pos}. This objective ensures that the GA evolves an ALC with the minimum hamming distance from the set of self patterns. This objective implies that the similarity between the evolved ALC's receptor and the set of self patterns needs to be maximised. The similarity between two patterns is calculated by the complement of their hamming distance, i.e. $k - a_{pos}$ where k is the length of the receptor and a_{pos} the ADT.

The fitness function υ_{pos} for positive selection is defined as

$$\upsilon_{pos} = w_1 (k - a_{pos}) + w_2 \chi(D, \mathbf{r}) \tag{6.2}$$

where w_1 and w_2 have the same meaning as for the adapted negative selection.

6.1.4 Parent Selection

The chromosomes that are selected for crossover to produce a set of offspring is randomly selected from an elitist set of chromosomes (elitism is discussed in section 3.3.3.). The generation gap is calculated as

$$\varsigma = I * e$$

where e is the rate of elitism, with $e \in (0,1]$.

A small value of e results in more diversity among the individuals for the next generation. If e is too big there will be less diversity among individuals for the next generation. After the elitist set has been created, the GA applies uniform crossover (as explained in section 3.3.1.2) between the randomly selected parents with probability, p_c, to produce offspring. The produced offspring forms part of the offspring set for the current generation. The crossover probability, p_c, decreases with an increase in the number of generations, i.e.

$$p_c = 1.0 - \frac{g}{G}$$

where g is the completed number of generations and G is the maximum allowed number of generations. With initial value $g = 0$, p_c will have an initial value equal to 1.0, implying a high probability of crossover. As the population evolves and becomes fitter, the need to exchange genetic material between fit individuals decreases. Thus the probability of crossover decreases as the population evolves.

6.1.5 Mutation

The offspring created through the uniform crossover operator (as explained above), is randomly selected for mutation. The GA applies random mutation (as explained in section 3.3.2.2) on the selected individual with probability, p_m. The probability, p_m, is calculated for each selected individual using

$$p_m = \frac{k - \chi(D, \mathbf{r})}{k}$$

where k is the length of the ALC's receptor and $\chi(D, \mathbf{r})$ is the average hamming distance between the ALC and the existing set of ALCs. $\frac{k-\chi(D,\mathbf{r})}{k}$ gives the average similarity rate between the ALC's receptor and the existing set of ALCs. Since one of the objectives is to ensure that the GA evolves an ALC with the lowest average overlap with an existing set of ALCs, the similarity rate should determine p_m. A high similarity rate implies that the ALC needs to be mutated with a high probability (mutation rate) to evolve a mutated ALC with the lowest average overlap with an existing set of ALCs. A low similarity rate implies that the ALC has a low average overlap with an existing set of ALCs and thus the rate of mutation must be low.

6.1.6 Selection of the New Population

The new population is selected from the parents of the current population and the offspring. All individuals (parents) in the elitist set survive to the next generation to ensure that the maximum fitness in the population does not decrease. The remainder of the new population is filled with the fittest offspring, determined using linear ranking (as explained in section 3.3.3).

6.1.7 Convergence of the GA

In the above GA algorithm, convergence in step 4 is reached when the maximum number of generations is reached or when the difference in the 2-point moving average between the fitness of the new population and the fitness of the previous generation's population is less than μ_T. The 2-point moving average is calculated as follows

$$\mu_g = \frac{\mu(H_{g-1}) + \mu(H_g)}{2.0}$$

where $\mu(H_{g-1})$ is the average fitness of population H_{g-1}, $\mu(H_g)$ is the average fitness of population H_g, and g is the current generation number. μ_g is calculated after each generation. The population H_g has converged if $|\mu_{g-1} - \mu_g| < \mu_T$ for $g = 1, ..., WindowSize$. A small difference in moving average, i.e. less than μ_T, implies that the reproduction and selection operators on the population H_g resulted in a minor change on the average fitness of the population and thus further evolution is unnecessary.

6.2 The GAIS Algorithm

This section explains the classification process of GAIS. The GA in phase 1 is repeatedly applied until the evolved active ALC set has converged (convergence of the active ALC set is explained in section 6.2.2). The ALCs in the converged active set, D_L, are then trained with one of the selection methods (as explained in section 5.2.2). After training, the *EMR* is set to 0.0 and GAIS tries to detect a set of patterns equal to the iteration size, *IS*, as non-self patterns using D_L. When a pattern is detected as non-self, the hit counter, *HC*, of each ALC that detected the non-self pattern is updated (as explained in section 5.2.3). If a pattern is not detected as non-self, the ALCs in the annihilated set is used to find a match. If a match is found, the annihilated ALC becomes mature and is moved to D_L. After all the patterns in an iteration have been classified, the life counter, $\tau(IS)$, of each ALC in the active set D_L is calculated to determine in which state an ALC

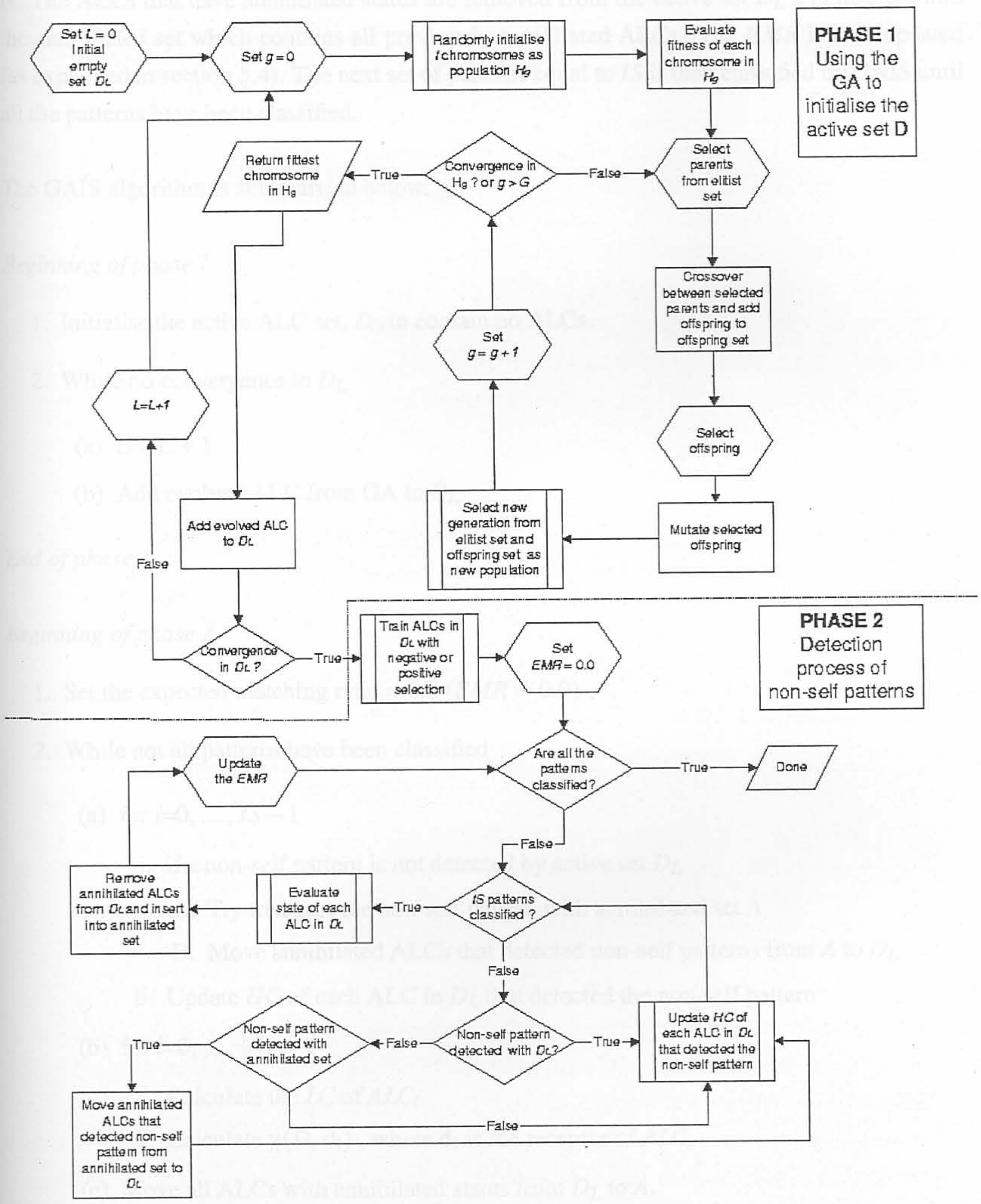

Set $L=0$ Initial empty set D_L
Set $g=0$
Randomly initialise I chromosomes as population H_g
Evaluate fitness of each chromosome in H_g
PHASE 1 Using the GA to initialise the active set D
Return fittest chromosome in H_g
True
Convergence in H_g ? or $g>G$
False
Select parents from elitist set
Crossover between selected parents and add offspring to offspring set
Set $g=g+1$
$L=L+1$
Select offspring
Select new generation from elitist set and offspring set as new population
Mutate selected offspring
Add evolved ALC to D_L
False
PHASE 2 Detection process of non-self patterns
Convergence in D_L ?
True
Train ALCs in D_L with negative or positive selection
Set $EMR=0.0$
Update the EMR
Are all the patterns classified?
True
Done
False
Remove annihilated ALCs from D_L and insert into annihilated set
Evaluate state of each ALC in D_L
True
IS patterns classified ?
False
Non-self pattern detected with annihilated set
False
Non-self pattern detected with D_L?
True
Update HC of each ALC in D_L that detected the non-self pattern
True
Move annihilated ALCs that detected non-self pattern from annihilated set to D_L
False

Figure 6.1: Flow layout of the Artificial Immune System

is. The ALCs that have annihilated status are removed from the active set D_L and inserted into the annihilated set which contains all previously annihilated ALCs. The *EMR* is then updated (as explained in section 5.4). The next set of patterns equal to *IS* is then classified by GAIS until all the patterns have been classified.

The GAIS algorithm is summarised below:

```
Beginning of phase 1

1. Initialise the active ALC set, $D_0$ to contain no ALCs.

2. While no convergence in $D_L$

    (a) L = L + 1

    (b) Add evolved ALC from GA to $D_L$

End of phase 1

Beginning of phase 2

1. Set the expected matching ratio = 0.0 ($EMR = 0.0$)

2. While not all patterns have been classified

    (a) for $i$=0, ...., $IS - 1$

        i. if a non-self pattern is not detected by active set $D_L$

            A. Try to detect the non-self pattern with annihilated set $A$

            B. Move annihilated ALCs that detected non-self patterns from $A$ to $D_L$

        ii. Update $HC$ of each ALC in $D_L$ that detected the non-self pattern

    (b) for $l$=0, ...., $|D_L|$ - 1

        i. Calculate the $LC$ of $ALC_l$

        ii. Calculate $\chi(D, \mathbf{d}_l)$ , where $\mathbf{d}_l$ is the receptor of $ALC_l$

    (c) Move all ALCs with annihilated status from $D_L$ to $A$.

    (d) Update the $EMR$ (as explained in section 5.4)
```

End of phase 2

Several aspects of the above algorithm need to be explained in more detail, for example the initialisation process of D in phase 1 and the convergence of D in phase 1, step 2. These aspects are discussed in the following sections.

6.2.1 Initialisation of the ALC Set

The initial ALC set is defined as the empty set $D_0 = \{\}$. $|D|$ increases after the best ALC as determined by the GA is added to D, i.e. $D_L = D_{L-1} \cup \{ALC_l\}$ where ALC_l is the evolved ALC and $l = 0, \ldots\ldots, L$. L is the number of ALCs evolved by the GA before the active ALC set converged. For each $l > 0$, the existing active set D_L is used by the fitness function to ensure that the newly evolved ALC_l is on average the furthest away from the existing active ALC set D_L.

6.2.2 Convergence of the ALC Set

In the above GAIS algorithm, convergence in phase 1, step 2 is reached when the difference in the 2-point moving average between the fitness of D_{L-1} and D_L is less then μ_T. The 2-point moving average is calculated as follows

$$\mu_L = \frac{\mu(D_{L-1}) + \mu(D_L)}{2.0}$$

where $\mu(D_{L-1})$ is the average fitness of set D_{L-1} and $\mu(D_L)$ is the average fitness of set D_L. μ_L is calculated after each evolved ALC is added to D_{L-1}. The set D_L has converged if $|\mu_{l-1} - \mu_l| < \mu_T$ for $l = L - WindowSize, \ldots., L$. After the ALC set has converged, all the ALCs in the active set are trained with one of the selection methods (as explained in section 5.2.2).

6.3 Conclusion

This chapter showed how a GA can be used to evolve an optimal set of ALCs using the proposed operators. The initialisation of a population, the selection operator, reproduction operators and the fitness function of the GA were discussed. The GAIS algorithm was presented and the initialisation of the ALC set were discussed. There are many other operators that can be considered when implementing the GA (as discussed in chapter 3), but the chosen operators have satisfying results (presented in the next chapter). Finding the best operators for the evolutionary process falls outside the scope of this dissertation. The focus is on classifying non-self patterns from seen self patterns with a set of ALCs that consists solely of ALCs with mature and memory status. The next chapter presents and analyses the results obtained from the GAIS classifier.

Chapter 7

Experimental Results

"No amount of experimentation can ever prove me right,
a single experiment can prove me wrong"
- Albert Einstein

This chapter presents experimental results to analyse the behavior of the GAIS model on different classification problems. The performance of GAIS is investigated under different control parameter values. The data sets were collected from the UCI Machine Learning Repository [7]. The patterns in each data set were discretised and converted to binary strings (as explained in section 6.2.1). For each experiment, one of the classes in a data set is selected as the self set. The self set is then used to train the ALCs with the adapted negative selection and the positive selection methods. The other classes in the data set represent the non-self patterns.

All experiments used a 30-fold cross validation self set. The self set was randomly divided into thirty disjoint sets. The ALCs were trained on 29 of these self sets and were tested with a test set that consisted of the remaining self set (the training set that was left out during training) and the unseen non-self patterns. In each experiment the initial population size in the GA was set to 100 chromosomes ($I = 100$) and the rate of elitism was set to 30% ($e = 0.3$). The window size in the GA and GAIS was set to 4.0 ($WindowSize = 4.0$) and $\mu_T = 0.01$ to test for convergence in the population and ALC set respectively. These values were found empirically to deliver good performance. All experimental results in this chapter are averages over 30 simulations with the selected self class in *italic* print. The best parameter settings are printed in **bold** in each table.

The results in each table, starting with the leftmost column, are the iteration size (IS) and parameter $w_1 * 100\%$ (W1) in the fitness function of the GA (recall that $w_2 = 1.0 - w_1$). The following results are the averages after the ALC set in GAIS has converged and all patterns were classified: the average number of ALCs in the active set (#ALCs), the average number of ALCs in the active set with memory status (#MemALCs), the average number of false positives (#fPos), the average number of false negatives (#fNeg), the average affinity distance threshold in the active set (ADT) and the average hamming distance between ALCs in the active set (HD). The average HD indicates the average hamming seperation among ALCs in the active set to cover non-self space. A higher value of HD indicates less overlap. The standard deviation is given in parentheses. A pattern from the self class that is falsely classified as non-self is referred to as a *false positive*. A pattern that is not from the self class and is falsely classified as self is referred to as a *false negative*. The average number of misclassified patterns for each parameter setting in each table is calculated as follows:

$$\#Misclassified = falsePositives + falseNegatives$$

Note that the accuracy is the average over all iterations. The interval-values for IS were calculated by

$$IS = \frac{is}{100} * \text{Size of data set}, is \in [25, 50, 75, 100]$$

The total number of iterations that GAIS executes is therefore equivalent to $\frac{\text{Size of the data set}}{IS}$. The selected values for w_1 were calculated by

$$w_1 = \frac{W1}{100}, W1 \in [25, 50, 75, 100]$$

Some of the tables are accompanied by figures that illustrate the results with the best parameter settings. These figures are the average number of ALCs in the active set, the average fitness of the ALCs in the active set and/or the average number of misclassified patterns at each iteration of GAIS. The figure that illustrates the average number of misclassified patterns, illustrates the average number of misclassified patterns after classifying *IS* patterns at each iteration. The number of patterns that was classified in the last iteration is less or equal to *IS*, i.e. the remaining patterns in the test set. The sum of the average number of misclassified patterns per iteration will correspond to the calculated number of misclassifications for the specific parameter settings in the corresponding tables.

Section 7.1 to section 7.5 discuss the results obtained from the GAIS model to classify the classes in the Iris data set, Wisconsin breast cancer data set, Mushroom data set, Glass data set, and the Car evaluation data set respectively with different parameter settings. The results of each of these data sets are concluded in their respective sections and a comparison with C4.5 is given in section 7.6.

7.1 Iris

The iris data set contains three classes of fifty instances each, where each class refers to a type of iris plant. The setosa class is linearly separable from the versicolor class and the virginica class. The versicolor class and the virginica class are not linearly separable. The dataset consists of 150 patterns, evenly distributed among the three classes (33.3% each). Each pattern consists of four continuously valued attributes. The patterns were converted to binary strings of length 20.

7.1.1 Setosa

As a first experiment, setosa was selected as self. The results for training the ALCs with negative selection on patterns of the setosa class as self after convergence are summarised in table 7.1. Most of the parameter settings in table 7.1 had an average false negative classification of 0.000. The overall best result among the parameter settings with 0.000 false negative classification, is with IS=37 and W1=50 since the average number of ALCs in the active set of ALCs (#ALCs = 16.427) and the false positive classification (fPos = 0.500) are the lowest. This gives a number of 0.500 patterns misclassified ($\#Misclassified = 0.500 + 0.000 = 0.500$) and a correct classification of 99.666%. The number of ALCs with memory status in the active set was on average 12.980. Figure 7.1 shows that the active set of ALCs started with an initial average size of 16.566 ALCs in the active set and then decreased over five iterations to an average number of 16.233 ALCs in the active set. The average fitness of the ALC set decreased over the iterations from 9.452 to 9.428. There was a decrease in misclassification over the iterations with an increase at iteration four.

Table 7.2 summarises the results for training the ALCs with positive selection on patterns from the setosa class as self. The positive selection also has different parameter settings for which a false negative classification of 0.000 was obtained. When IS=37 and W1=50 the lowest misclassification was achieved ($\#Misclassified = 0.567 + 0.000 = 0.567$) with the least average number of ALCs in the active set of ALCs (#ALCs = 17.420). This gives a correct classification of 99.622%. The average number of ALCs with memory status in the active set of ALCs

Table 7.1: *Setosa* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	20.067 (±6.285)	14.727 (±5.004)	0.767 (±1.995)	0.167 (±0.531)	9.423 (±0.139)	9.393 (±0.213)
	50	**16.427 (±5.187)**	**12.980 (±4.203)**	**0.500 (±1.306)**	**0.000 (±0.000)**	**10.874 (±0.180)**	**8.012 (±0.188)**
	75	21.900 (±4.664)	17.520 (±3.731)	0.633 (±2.189)	0.033 (±0.183)	11.946 (±0.131)	6.138 (±0.253)
	100	26.667 (±9.932)	21.333 (±7.946)	0.567 (±1.478)	0.133 (±0.434)	12.099 (±0.110)	5.191 (±0.311)
74	25	19.511 (±5.504)	12.000 (±3.614)	0.767 (±1.995)	0.233 (±0.626)	9.328 (±0.167)	9.518 (±0.097)
	50	17.844 (±3.963)	11.711 (±2.652)	0.633 (±1.829)	0.000 (±0.000)	10.929 (±0.200)	8.011 (±0.191)
	75	23.567 (±5.117)	15.711 (±3.411)	0.667 (±1.826)	0.000 (±0.000)	11.946 (±0.178)	6.164 (±0.273)
	100	23.733 (±12.273)	15.822 (±8.182)	0.633 (±2.008)	0.567 (±1.135)	12.102 (±0.123)	5.148 (±0.377)
112	25	20.933 (±6.147)	9.883 (±3.183)	0.733 (±1.999)	0.300 (±0.702)	9.182 (±0.215)	9.616 (±0.057)
	50	16.933 (±4.017)	8.400 (±2.061)	0.567 (±1.478)	0.000 (±0.000)	10.878 (±0.163)	8.035 (±0.178)
	75	23.233 (±6.207)	11.617 (±3.104)	0.767 (±2.176)	0.000 (±0.000)	11.946 (±0.139)	6.131 (±0.209)
	100	27.400 (±10.627)	13.700 (±5.314)	0.500 (±1.480)	0.267 (±1.048)	12.107 (±0.172)	5.187 (±0.354)
150	25	20.567 (±4.636)	0.000 (±0.000)	0.733 (±1.999)	0.067 (±0.254)	9.193 (±0.131)	9.618 (±0.052)
	50	17.433 (±3.540)	0.000 (±0.000)	0.733 (±2.363)	0.000 (±0.000)	10.895 (±0.184)	8.013 (±0.173)
	75	21.667 (±5.616)	0.000 (±0.000)	0.700 (±1.643)	0.000 (±0.000)	11.956 (±0.154)	6.138 (±0.268)
	100	23.433 (±11.820)	0.000 (±0.000)	0.467 (±1.479)	0.400 (±0.724)	12.083 (±0.129)	5.151 (±0.352)

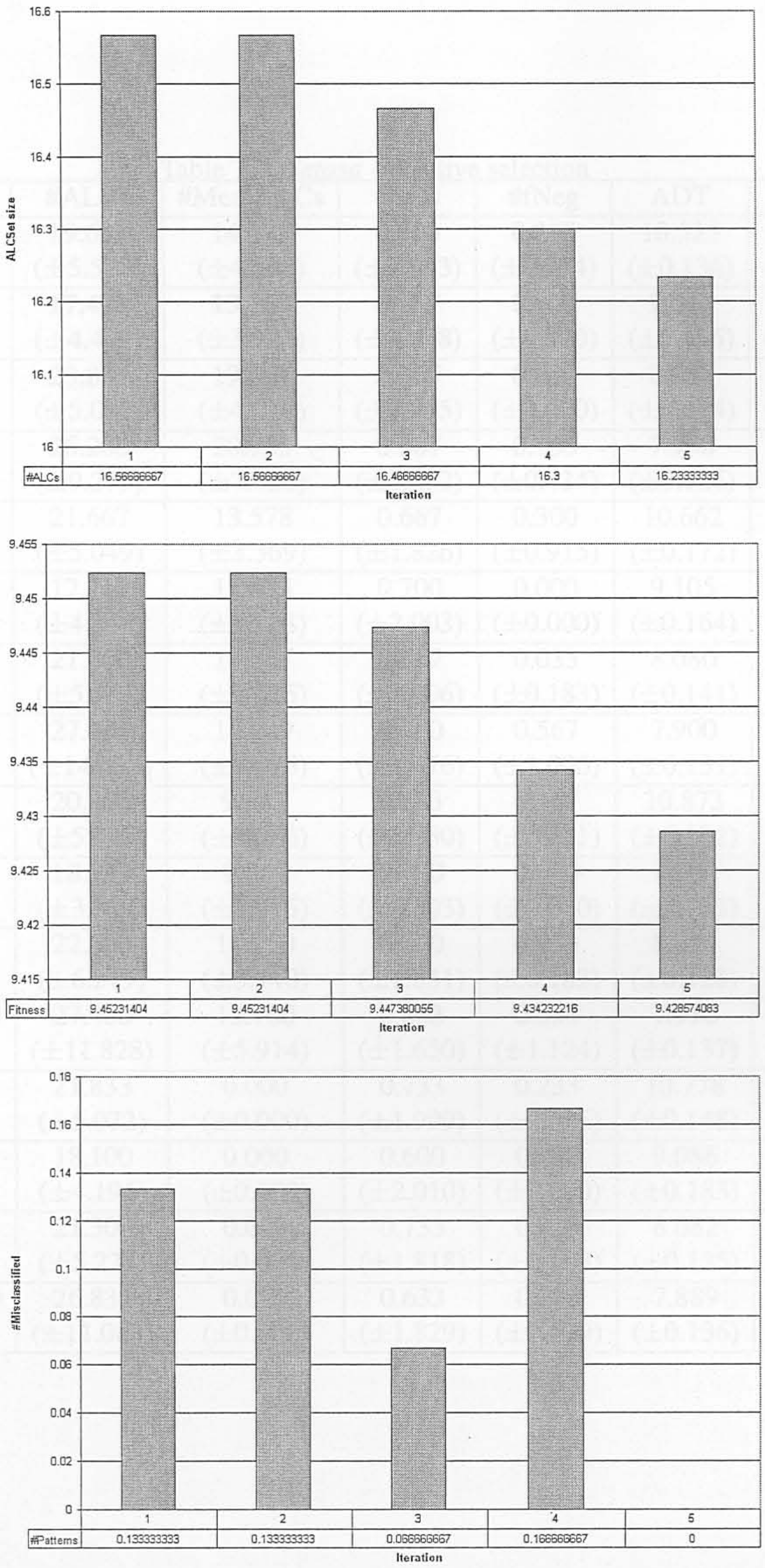


Figure 7.1: *Setosa* - Negative selection with IS=37 and W1=50

Table 7.2: *Setosa* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	19.653 (±5.572)	14.433 (±4.506)	0.700 (±1.643)	0.233 (±0.774)	10.525 (±0.136)	9.387 (±0.176)
	50	**17.420 (±4.409)**	**13.760 (±3.524)**	**0.567 (±1.478)**	**0.000 (±0.000)**	**9.043 (±0.196)**	**7.948 (±0.209)**
	75	23.833 (±5.093)	19.067 (±4.074)	0.767 (±1.995)	0.000 (±0.000)	8.087 (±0.154)	6.212 (±0.277)
	100	26.200 (±9.279)	20.953 (±7.426)	0.567 (±1.832)	0.133 (±0.434)	7.900 (±0.126)	5.183 (±0.384)
74	25	21.667 (±5.049)	13.578 (±3.369)	0.667 (±1.826)	0.300 (±0.915)	10.662 (±0.172)	9.546 (±0.069)
	50	17.711 (±4.846)	11.633 (±3.308)	0.700 (±2.003)	0.000 (±0.000)	9.105 (±0.164)	8.038 (±0.121)
	75	21.700 (±5.453)	14.467 (±3.635)	0.667 (±2.006)	0.033 (±0.183)	8.080 (±0.141)	6.214 (±0.274)
	100	27.133 (±14.063)	18.089 (±9.375)	0.400 (±1.476)	0.567 (±1.006)	7.900 (±0.131)	5.044 (±0.468)
112	25	20.400 (±5.090)	9.533 (±2.566)	0.833 (±2.350)	0.167 (±0.531)	10.873 (±0.192)	9.641 (±0.047)
	50	18.333 (±3.889)	9.067 (±1.915)	0.700 (±2.003)	0.000 (±0.000)	9.102 (±0.153)	8.037 (±0.185)
	75	22.500 (±6.279)	11.250 (±3.140)	0.600 (±1.831)	0.033 (±0.183)	8.071 (±0.128)	6.085 (±0.303)
	100	27.400 (±11.828)	13.700 (±5.914)	0.633 (±1.650)	0.333 (±1.124)	7.910 (±0.137)	5.236 (±0.311)
150	25	21.833 (±5.072)	0.000 (±0.000)	0.733 (±1.999)	0.233 (±0.626)	10.778 (±0.148)	9.615 (±0.042)
	50	18.100 (±4.196)	0.000 (±0.000)	0.600 (±2.010)	0.000 (±0.000)	9.086 (±0.183)	8.060 (±0.203)
	75	21.300 (±5.227)	0.000 (±0.000)	0.733 (±1.818)	0.033 (±0.183)	8.082 (±0.125)	6.187 (±0.218)
	100	26.833 (±11.083)	0.000 (±0.000)	0.633 (±1.829)	0.233 (±0.679)	7.889 (±0.136)	5.136 (±0.313)

is 13.760. Figure 7.2 shows that the active set of ALCs had an initial average size of 17.6 and decreased over five iterations to an average number of 17.266. The average fitness also decreased over the iterations from 9.458 to 9.446 and the average misclassification increased from iteration one to two, and then decreased to iteration five.

7.1.2 Versicolor

The classification results for training the ALCs with negative selection on patterns of the versicolor class as self are shown in table 7.3. The lowest misclassification ($\#Misclassified = 0.967 + 3.333 = 4.300$) was achieved with IS=37 and W1=75. This gives a correct classification of 97.133% with an average number of 21.567 ALCs in the active set of ALCs. The average number of ALCs with memory status in the active set was 17.220. Figure 7.3 shows that their was no change in the size of the active set over all iterations and the constant size is an average number of 21.566 ALCs. The average fitness of the ALC set was constant at 9.908 and the average number of misclassification increased from iteration one to iteration three and then decreased to iteration five.

The best classification result shown in table 7.4, is achieved with IS=37 and W1=50 when training the ALCs with positive selection on the patterns of the versicolor as self. The misclassification of 4.5 patterns ($\#Misclassified = 1.167 + 3.333 = 4.5$) gives a correct classification of 97.000%. The average number of ALCs in the active set of ALCs was 20.333 and an average number of 16.253 of these had memory status. Figure 7.4 shows that the average number of ALCs in the active set of ALCs was constant at 20.333 over all iterations. The average number of misclassification increased from iteration one to three and then decreased to iteration five.

7.1.3 Virginica

Table 7.5 shows that with IS=150 and W1=75 the lowest misclassification of 8.100 patterns ($\#Misclassified = 1.200 + 6.900 = 8.100$) was achieved when training the ALCs with negative selection on patterns of the virginica class as self. This gives a correct classification of 94.600% with an average number of 23.533 ALCs in the active set. An average number of 0.000 of the ALCs in the active set had memory status, since $IS = 150$ (which is the size of the data set) which implies that there was only one iteration. Since an ALC's status is evaluated after an iteration, the active set of ALCs in the first and only iteration cannot contain ALCs with memory status.

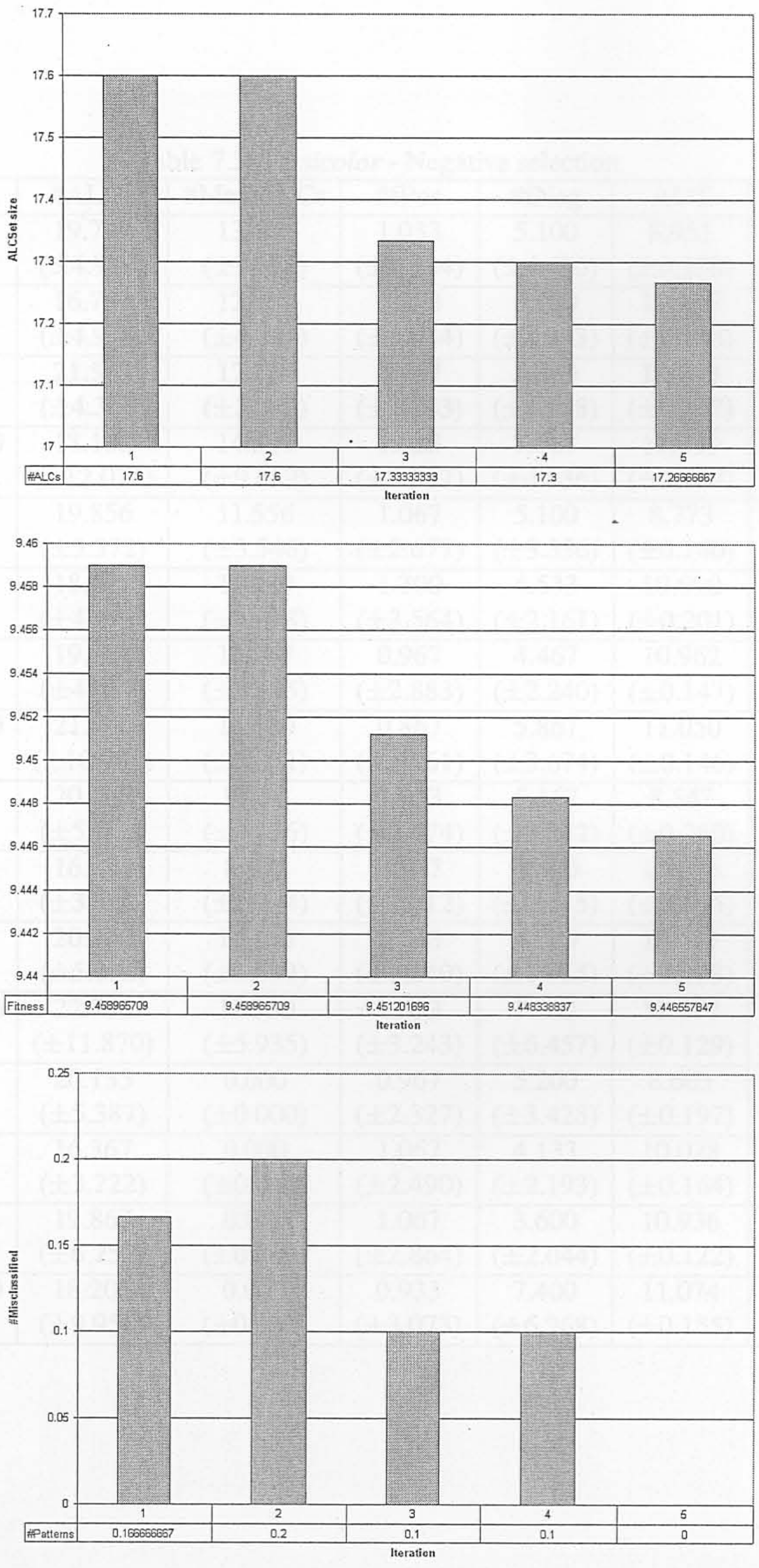


Figure 7.2: *Setosa* - Positive selection with IS=37 and W1=50

Table 7.3: *Versicolor* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	19.747 (±4.979)	13.660 (±3.953)	1.033 (±2.684)	5.100 (±4.436)	8.951 (±0.158)	9.454 (±0.148)
	50	16.753 (±4.979)	12.887 (±4.001)	1.233 (±3.014)	5.000 (±2.983)	10.105 (±0.148)	8.315 (±0.187)
	75	**21.567 (±4.337)**	**17.220 (±3.483)**	**0.967 (±2.883)**	**3.333 (±1.918)**	**10.943 (±0.117)**	**6.805 (±0.243)**
	100	18.133 (±12.074)	14.507 (±9.659)	1.033 (±3.057)	7.667 (±4.436)	11.049 (±0.117)	5.666 (±0.476)
74	25	19.856 (±5.372)	11.556 (±3.546)	1.067 (±2.677)	5.100 (±3.356)	8.773 (±0.140)	9.587 (±0.098)
	50	18.422 (±4.239)	11.989 (±2.828)	1.300 (±3.564)	4.533 (±2.161)	10.090 (±0.201)	8.407 (±0.168)
	75	19.900 (±4.957)	13.267 (±3.305)	0.967 (±2.883)	4.467 (±2.240)	10.962 (±0.147)	6.799 (±0.283)
	100	21.600 (±10.981)	14.400 (±7.321)	0.867 (±2.161)	5.867 (±3.674)	11.050 (±0.146)	5.877 (±0.313)
112	25	20.267 (±5.717)	9.250 (±3.036)	1.133 (±2.474)	5.167 (±3.842)	8.547 (±0.250)	9.708 (±0.042)
	50	16.867 (±3.721)	8.367 (±1.934)	0.967 (±2.512)	4.300 (±2.215)	10.073 (±0.165)	8.364 (±0.172)
	75	20.200 (±5.006)	10.100 (±2.503)	1.133 (±2.849)	4.000 (±1.965)	10.947 (±0.133)	6.850 (±0.229)
	100	22.700 (±11.870)	11.350 (±5.935)	1.033 (±3.243)	7.033 (±6.457)	11.077 (±0.129)	5.587 (±0.442)
150	25	20.133 (±5.387)	0.000 (±0.000)	0.967 (±2.327)	5.200 (±3.428)	8.603 (±0.197)	9.708 (±0.041)
	50	16.367 (±3.222)	0.000 (±0.000)	1.067 (±2.490)	4.133 (±2.193)	10.028 (±0.164)	8.440 (±0.134)
	75	19.867 (±6.252)	0.000 (±0.000)	1.067 (±2.864)	3.600 (±2.044)	10.936 (±0.122)	6.859 (±0.252)
	100	18.200 (±9.956)	0.000 (±0.000)	0.933 (±3.073)	7.400 (±6.268)	11.074 (±0.155)	5.628 (±0.501)

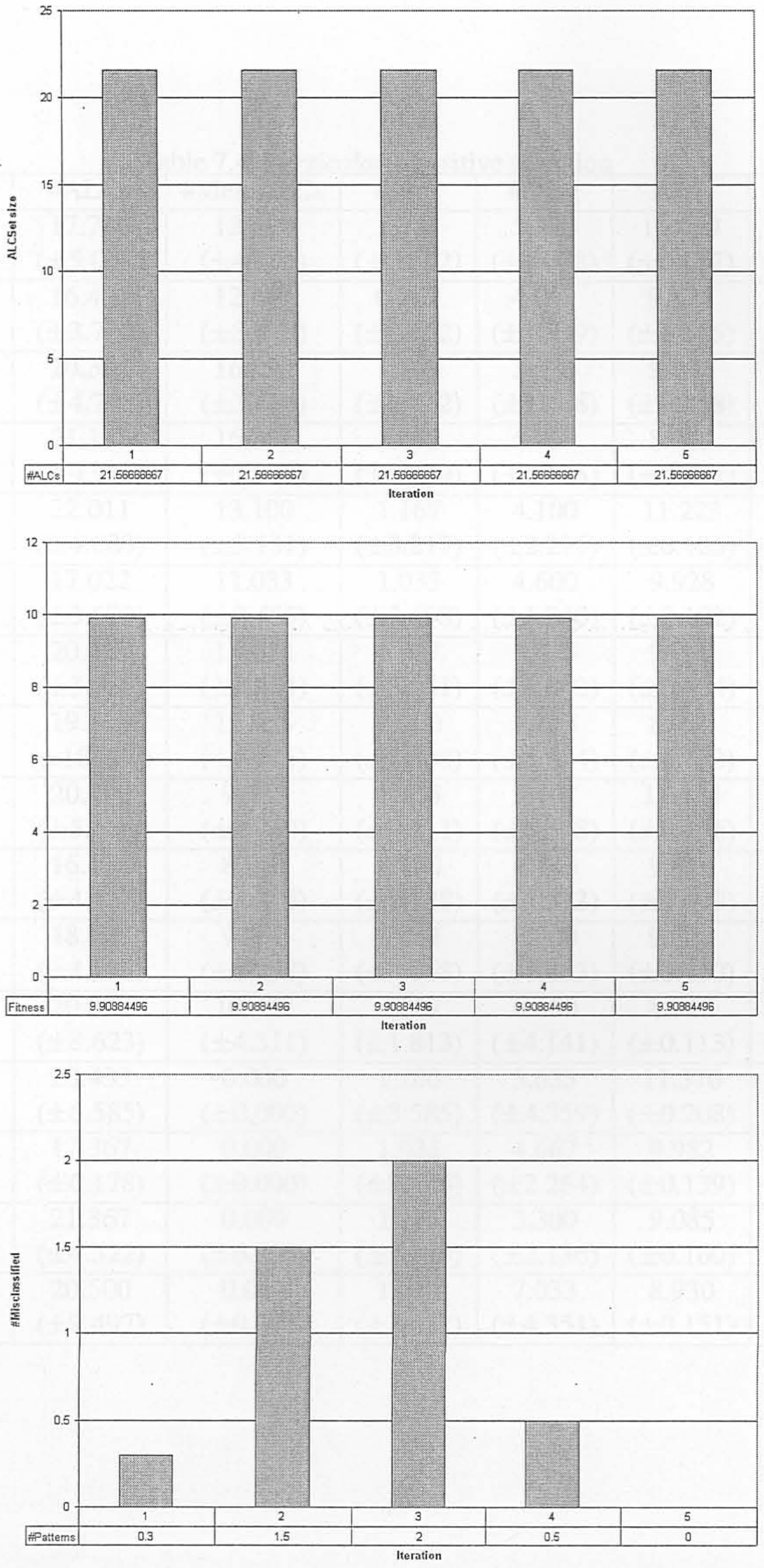


Figure 7.3: *Versicolor* - Negative selection with IS=37 and W1=75

Table 7.4: *Versicolor* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	17.780 (±5.001)	12.160 (±4.066)	1.133 (±2.662)	5.300 (±4.078)	11.039 (±0.177)	9.348 (±0.330)
	50	16.473 (±3.780)	12.607 (±3.018)	0.967 (±2.512)	4.067 (±1.999)	9.875 (±0.165)	8.277 (±0.226)
	75	**20.333 (±4.700)**	**16.253 (±3.750)**	**1.167 (±2.842)**	**3.333 (±1.768)**	**9.041 (±0.118)**	**6.757 (±0.256)**
	100	21.133 (±9.916)	16.887 (±7.942)	1.033 (±3.429)	6.067 (±4.025)	8.919 (±0.137)	5.784 (±0.400)
74	25	22.011 (±4.683)	13.100 (±3.131)	1.167 (±3.217)	4.100 (±2.295)	11.223 (±0.125)	9.624 (±0.083)
	50	17.022 (±3.529)	11.033 (±2.455)	1.033 (±2.498)	4.600 (±1.940)	9.928 (±0.192)	8.410 (±0.173)
	75	20.433 (±3.636)	13.622 (±2.424)	1.067 (±3.051)	3.833 (±1.642)	9.070 (±0.134)	6.802 (±0.257)
	100	19.133 (±10.224)	12.756 (±6.816)	0.900 (±2.708)	7.233 (±5.104)	8.951 (±0.123)	5.601 (±0.488)
112	25	20.700 (±5.676)	9.483 (±2.740)	1.133 (±3.411)	5.467 (±5.178)	11.404 (±0.196)	9.710 (±0.038)
	50	16.433 (±4.523)	8.050 (±2.175)	0.900 (±1.788)	4.433 (±2.373)	9.946 (±0.179)	8.417 (±0.205)
	75	18.600 (±4.515)	9.300 (±2.258)	0.967 (±3.068)	3.800 (±1.883)	9.022 (±0.150)	6.743 (±0.283)
	100	20.833 (±8.623)	10.417 (±4.311)	0.767 (±1.813)	6.600 (±4.141)	8.945 (±0.113)	5.823 (±0.377)
150	25	20.433 (±6.585)	0.000 (±0.000)	1.200 (±3.585)	5.633 (±4.359)	11.376 (±0.208)	9.685 (±0.049)
	50	17.367 (±6.178)	0.000 (±0.000)	1.033 (±2.684)	4.667 (±2.264)	9.982 (±0.139)	8.429 (±0.144)
	75	21.367 (±4.522)	0.000 (±0.000)	1.267 (±3.383)	3.300 (±2.136)	9.085 (±0.160)	6.894 (±0.284)
	100	20.500 (±9.497)	0.000 (±0.000)	1.033 (±3.057)	7.033 (±4.351)	8.930 (±0.151)	5.784 (±0.370)

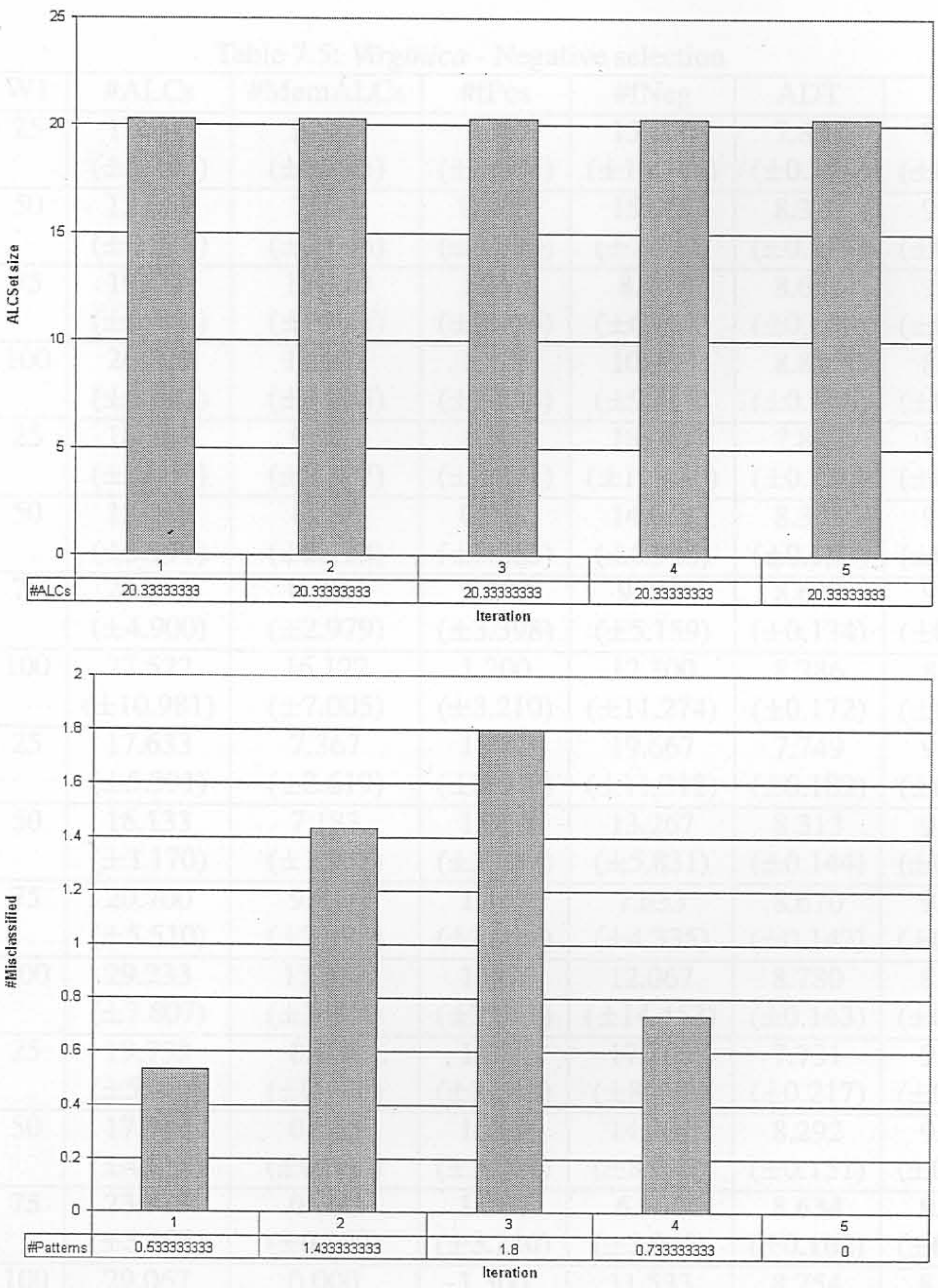


Figure 7.4: *Versicolor* - Positive selection with IS=37 and W1=75

Table 7.5: *Virginica* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	17.293 (±5.395)	9.873 (±3.835)	1.067 (±3.051)	15.000 (±10.342)	7.859 (±0.155)	9.604 (±0.249)
	50	13.667 (±3.779)	7.973 (±2.565)	0.967 (±2.883)	15.967 (±7.327)	8.327 (±0.189)	9.271 (±0.214)
	75	19.420 (±6.566)	12.920 (±4.881)	1.300 (±3.186)	8.467 (±6.962)	8.688 (±0.144)	9.020 (±0.278)
	100	26.527 (±6.502)	17.993 (±4.725)	1.133 (±3.224)	10.800 (±9.845)	8.814 (±0.156)	8.055 (±0.414)
74	25	18.289 (±4.930)	9.222 (±2.807)	1.100 (±3.231)	18.067 (±10.110)	7.812 (±0.150)	9.823 (±0.068)
	50	15.767 (±3.177)	8.356 (±2.192)	0.967 (±3.253)	14.933 (±6.913)	8.323 (±0.137)	9.466 (±0.117)
	75	20.811 (±4.900)	11.911 (±2.979)	1.200 (±3.398)	9.267 (±5.159)	8.672 (±0.134)	9.082 (±0.250)
	100	27.522 (±10.981)	16.122 (±7.005)	1.200 (±3.210)	12.300 (±11.274)	8.786 (±0.172)	8.059 (±0.383)
112	25	17.633 (±5.391)	7.367 (±2.619)	1.000 (±2.505)	19.667 (±11.312)	7.749 (±0.182)	9.881 (±0.038)
	50	16.133 (±3.170)	7.183 (±1.517)	1.067 (±3.237)	13.267 (±5.831)	8.313 (±0.144)	9.559 (±0.088)
	75	20.700 (±5.510)	9.817 (±2.490)	1.167 (±3.217)	7.633 (±4.335)	8.670 (±0.142)	9.155 (±0.158)
	100	29.233 (±7.807)	13.333 (±3.724)	1.067 (±3.051)	12.067 (±14.453)	8.780 (±0.143)	8.079 (±0.428)
150	25	19.733 (±5.401)	0.000 (±0.000)	1.033 (±3.243)	17.200 (±8.438)	7.731 (±0.217)	9.896 (±0.029)
	50	17.333 (±4.054)	0.000 (±0.000)	1.100 (±3.044)	14.067 (±8.952)	8.292 (±0.151)	9.564 (±0.065)
	75	**23.533 (±5.198)**	**0.000 (±0.000)**	**1.200 (±3.210)**	**6.900 (±3.942)**	**8.634 (±0.162)**	**9.179 (±0.197)**
	100	29.067 (±7.061)	0.000 (±0.000)	1.300 (±3.375)	11.533 (±8.525)	8.754 (±0.143)	8.067 (±0.713)

Table 7.6: *Virginica* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
37	25	16.280 (±4.784)	9.207 (±3.385)	1.033 (±3.057)	16.767 (±11.548)	12.128 (±0.186)	9.605 (±0.181)
	50	13.373 (±3.344)	7.887 (±2.432)	1.067 (±2.864)	14.633 (±7.531)	11.620 (±0.163)	9.273 (±0.170)
	75	20.453 (±6.202)	13.367 (±4.763)	1.033 (±3.057)	9.767 (±7.890)	11.342 (±0.129)	9.113 (±0.207)
	100	24.087 (±7.786)	15.860 (±5.445)	1.033 (±3.057)	13.967 (±12.505)	11.207 (±0.168)	7.976 (±0.695)
74	25	18.011 (±5.426)	8.711 (±2.904)	0.900 (±2.893)	17.533 (±11.116)	12.192 (±0.157)	9.799 (±0.090)
	50	16.622 (±3.675)	9.056 (±2.370)	1.067 (±3.051)	14.600 (±7.166)	11.681 (±0.144)	9.520 (±0.092)
	75	20.178 (±5.535)	11.367 (±3.187)	1.100 (±2.857)	8.833 (±5.565)	11.375 (±0.143)	9.150 (±0.180)
	100	26.578 (±9.091)	15.211 (±5.456)	1.167 (±3.405)	14.033 (±13.265)	11.175 (±0.141)	7.933 (±0.525)
112	25	19.733 (±5.626)	8.267 (±2.605)	1.200 (±3.210)	16.833 (±11.946)	12.264 (±0.183)	9.890 (±0.038)
	50	15.633 (±3.168)	6.850 (±1.549)	1.167 (±3.030)	14.833 (±7.235)	11.695 (±0.169)	9.529 (±0.102)
	75	**23.833 (±4.793)**	**11.083 (±2.271)**	**1.267 (±3.194)**	**6.800 (±4.552)**	**11.342 (±0.127)**	**9.171 (±0.172)**
	100	28.000 (±8.208)	13.033 (±3.859)	1.167 (±3.030)	11.033 (±7.073)	11.234 (±0.165)	8.129 (±0.309)
150	25	21.700 (±5.292)	0.000 (±0.000)	1.100 (±2.857)	13.133 (±10.884)	12.195 (±0.158)	9.895 (±0.025)
	50	15.900 (±3.836)	0.000 (±0.000)	1.167 (±3.217)	14.367 (±5.295)	11.702 (±0.193)	9.557 (±0.093)
	75	20.400 (±5.184)	0.000 (±0.000)	1.167 (±3.030)	7.967 (±5.449)	11.337 (±0.167)	9.157 (±0.163)
	100	28.933 (±9.161)	0.000 (±0.000)	1.233 (±3.202)	11.500 (±10.875)	11.221 (±0.149)	8.105 (±0.325)

Table 7.6 shows that the best classification is achieved with IS=112 and W1=75 when training the ALCs with positive selection. These parameter settings gave the lowest misclassification of 8.067 patterns ($\#Misclassified = 1.267 + 6.800 = 8.067$) with an average number of 23.833 ALCs in the active set of ALCs that had an average of 11.083 ALCs with memory status. This gives a correct classification rate of 94.622%. The size of the active set of ALCs was constant on an average of 23.833 over time. The size of the active set of ALCs was constant on 23.833 over all iterations and the average fitness of the ALC set was constant.

7.1.4 Conclusion: Iris

With *setosa* or *versicolor* as the self class, training the ALCs with the negative selection method resulted in better classification than training with the positive selection method. When patterns of the *virginica* class was used as the self set the ALCs trained with positive selection had better classification than the ALCs trained with negative selection. There is also a decrease or constant average number of ALCs in the active set of ALCs for both negative and positive selection methods with the different classes as self. The decrease or constant average number of ALCs in the set results in the decrease or constant average fitness of the ALC set.

7.2 Wisconsin Breast Cancer

The Wisconsin breast cancer data set consists of 699 patterns that are distributed between 2 classes, namely benign and malignant. Each pattern consists of 9 attributes with values in the range [1,10]. The tenth attribute is the pattern's sample code number and uniquely identifies the pattern in the data set. The sample code number was left out in the training and testing of the AIS model. There are 16 missing attribute values for the bare nuclei attribute in the data set. 458 patterns are of the benign class and 241 patterns of the malignant class. The patterns were converted to binary strings of length 36. The missing values were represented by binary strings as straight 1's.

7.2.1 Benign

Table 7.7 shows the results for classifying the Wisconsin breast cancer data set with patterns of the benign class as the self set. The ALCs were trained with the negative selection method. The best classification result was achieved when IS=524 and W1=25. An average number of 34.567 ALCs formed part of the active set of ALCs. An average number of 16.867 of the ALCs in the

Table 7.7: *Benign* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
175	25	32.492 (±5.587)	22.367 (±4.342)	1.333 (±0.922)	7.400 (±7.859)	16.159 (±0.154)	17.417 (±0.102)
	50	21.858 (±5.735)	16.058 (±4.415)	1.133 (±0.973)	14.900 (±7.685)	18.052 (±0.195)	15.659 (±0.268)
	75	29.400 (±7.342)	22.033 (±5.521)	1.067 (±1.048)	9.633 (±3.746)	19.377 (±0.135)	13.516 (±0.254)
	100	37.633 (±12.936)	28.217 (±9.724)	1.200 (±1.031)	12.267 (±7.220)	19.764 (±0.121)	12.226 (±0.328)
350	25	33.400 (±5.190)	16.300 (±2.531)	1.233 (±1.006)	7.033 (±4.605)	16.060 (±0.134)	17.503 (±0.036)
	50	21.367 (±5.385)	10.583 (±2.678)	0.767 (±0.774)	13.167 (±6.422)	17.989 (±0.183)	15.786 (±0.186)
	75	27.500 (±7.960)	13.750 (±3.980)	0.933 (±0.740)	11.567 (±6.021)	19.384 (±0.171)	13.516 (±0.342)
	100	36.100 (±11.678)	18.050 (±5.839)	1.233 (±1.040)	11.267 (±7.012)	19.779 (±0.132)	12.271 (±0.492)
524	**25**	**34.567 (±5.029)**	**16.867 (±2.526)**	**1.367 (±1.189)**	**6.267 (±5.219)**	**16.061 (±0.188)**	**17.499 (±0.053)**
	50	20.967 (±5.021)	10.400 (±2.558)	1.000 (±0.871)	13.667 (±5.785)	17.963 (±0.163)	15.773 (±0.142)
	75	29.167 (±8.550)	14.583 (±4.275)	1.033 (±0.890)	9.933 (±5.675)	19.390 (±0.158)	13.508 (±0.270)
	100	39.167 (±11.908)	19.583 (±5.954)	0.933 (±0.785)	10.367 (±8.109)	19.795 (±0.156)	12.204 (±0.598)
699	25	33.100 (±6.467)	0.000 (±0.000)	1.400 (±0.932)	8.067 (±10.945)	16.023 (±0.233)	17.506 (±0.050)
	50	20.867 (±5.419)	0.000 (±0.000)	1.200 (±1.031)	12.433 (±5.643)	17.991 (±0.203)	15.764 (±0.211)
	75	27.933 (±8.440)	0.000 (±0.000)	0.933 (±0.868)	10.567 (±4.150)	19.390 (±0.136)	13.503 (±0.233)
	100	37.467 (±13.405)	0.000 (±0.000)	1.200 (±1.031)	11.700 (±7.274)	19.773 (±0.136)	12.193 (±0.443)

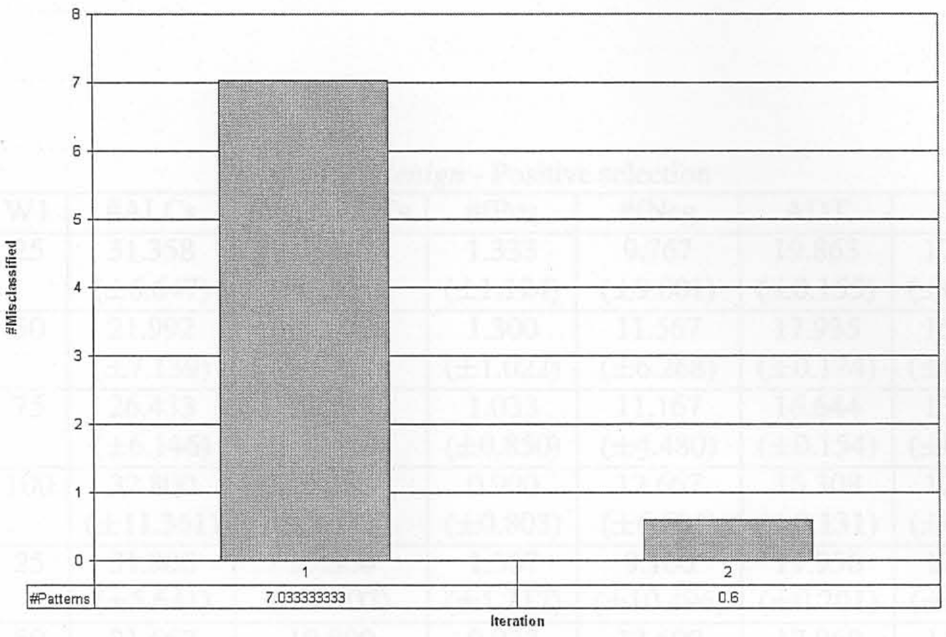


Figure 7.5: *Benign* - Negative selection with IS=524 and W1=25

active set had memory status. The ALCs classified 6.267 patterns falsely as benign and 1.367 as malignant. This gives a misclassification of 7.634 patterns ($\#Misclassified = 6.267 + 1.367 = 7.634$) and a correct classification of 98.907%. The size of the active set of ALCs was constant on 34.567 over all iterations and thus the average fitness of the ALC set was constant. Figure 7.5 shows a decrease in the average number of misclassification over the iterations.

Table 7.8 shows the results when the ALCs were trained with the positive selection method. The best classification result was achieved when IS=524 and W1=25. An average number of 33.800 ALCs formed part of the active set of ALCs. An average number of 16.467 of the ALCs in the active set had memory status. The ALCs classified 6.900 patterns falsely as benign and 1.500 as malignant which gave a misclassification of 8.400 patterns ($\#Misclassified = 6.900 + 1.500 = 8.400$) and a correct classification rate of 98.798%. The number of ALCs in the active set of ALCs was constant on 33.800 over all iterations with a constant average fitness.

7.2.2 Malignant

The results for training the ALCs with the negative selection method is shown in table 7.9. The patterns of the malignant class was used as the self set. The best classification result was

Table 7.8: *Benign* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
175	25	31.358 (±6.647)	21.442 (±5.110)	1.333 (±1.124)	9.767 (±9.001)	19.863 (±0.155)	17.401 (±0.089)
	50	21.992 (±7.139)	16.092 (±5.411)	1.300 (±1.022)	11.567 (±6.268)	17.935 (±0.174)	15.701 (±0.204)
	75	26.433 (±6.146)	19.792 (±4.614)	1.033 (±0.850)	11.167 (±4.480)	16.644 (±0.154)	13.530 (±0.324)
	100	32.800 (±11.361)	24.583 (±8.516)	0.900 (±0.803)	12.667 (±6.504)	16.308 (±0.131)	12.256 (±0.250)
350	25	31.900 (±5.641)	15.500 (±2.907)	1.367 (±1.217)	9.100 (±10.496)	19.956 (±0.201)	17.485 (±0.054)
	50	21.667 (±7.448)	10.800 (±3.732)	0.933 (±0.868)	13.600 (±8.046)	17.960 (±0.180)	15.694 (±0.185)
	75	28.600 (±6.667)	14.300 (±3.334)	1.267 (±0.907)	10.267 (±3.956)	16.622 (±0.145)	13.507 (±0.249)
	100	34.967 (±12.524)	17.483 (±6.262)	0.700 (±0.750)	11.867 (±6.942)	16.245 (±0.141)	12.262 (±0.321)
524	**25**	**33.800 (±3.398)**	**16.467 (±1.756)**	**1.500 (±1.075)**	**6.900 (±3.284)**	**19.922 (±0.117)**	**17.490 (±0.044)**
	50	21.867 (±5.619)	10.917 (±2.801)	1.133 (±0.937)	13.833 (±7.027)	18.001 (±0.149)	15.755 (±0.164)
	75	29.367 (±7.388)	14.683 (±3.694)	1.067 (±0.907)	10.000 (±5.675)	16.619 (±0.140)	13.429 (±0.312)
	100	38.133 (±12.875)	19.067 (±6.438)	1.200 (±1.031)	11.767 (±10.061)	16.214 (±0.207)	12.168 (±0.424)
699	25	32.900 (±6.989)	0.000 (±0.000)	1.600 (±1.404)	9.733 (±14.064)	19.958 (±0.258)	17.477 (±0.044)
	50	21.400 (±5.302)	0.000 (±0.000)	1.100 (±0.995)	11.467 (±5.710)	17.964 (±0.153)	15.710 (±0.183)
	75	27.200 (±7.179)	0.000 (±0.000)	1.000 (±0.983)	10.467 (±4.439)	16.637 (±0.175)	13.512 (±0.353)
	100	38.433 (±14.943)	0.000 (±0.000)	1.100 (±0.960)	11.500 (±8.228)	16.234 (±0.144)	12.125 (±0.451)

Table 7.9: *Malignant* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
175	25	23.342 (±3.960)	6.758 (±1.864)	2.633 (±1.189)	146.100 (±98.744)	13.923 (±0.162)	16.963 (±0.355)
	50	15.108 (±3.566)	5.650 (±1.666)	1.867 (±1.074)	98.533 (±69.563)	15.024 (±0.158)	15.835 (±0.420)
	75	19.133 (±4.832)	9.733 (±2.683)	1.833 (±1.367)	46.333 (±47.790)	15.954 (±0.138)	14.249 (±0.318)
	100	25.875 (±9.360)	13.975 (±5.936)	1.533 (±1.042)	62.200 (±71.823)	16.276 (±0.132)	12.756 (±0.663)
350	25	31.000 (±5.855)	7.867 (±2.432)	2.067 (±1.437)	117.633 (±95.248)	13.530 (±0.217)	17.739 (±0.038)
	50	21.033 (±6.856)	7.300 (±2.507)	1.967 (±1.326)	115.100 (±83.162)	14.842 (±0.159)	16.663 (±0.144)
	75	23.200 (±5.209)	9.500 (±2.338)	1.733 (±0.980)	46.500 (±24.221)	15.910 (±0.114)	14.664 (±0.234)
	100	34.900 (±13.510)	14.067 (±5.889)	1.700 (±1.393)	52.200 (±52.346)	16.271 (±0.139)	13.010 (±0.397)
524	25	30.800 (±6.885)	8.683 (±2.329)	2.300 (±1.418)	149.900 (±89.560)	13.563 (±0.165)	17.735 (±0.037)
	50	20.667 (±5.726)	7.750 (±2.586)	1.800 (±1.297)	108.633 (±89.016)	14.799 (±0.169)	16.710 (±0.188)
	75	25.133 (±6.709)	10.917 (±3.235)	1.967 (±1.129)	60.367 (±48.565)	15.918 (±0.130)	14.620 (±0.336)
	100	32.300 (±13.018)	14.133 (±5.810)	1.633 (±1.273)	49.833 (±54.484)	16.255 (±0.137)	13.002 (±0.463)
699	25	32.433 (±5.137)	0.000 (±0.000)	2.567 (±1.382)	112.500 (±68.027)	13.597 (±0.134)	17.732 (±0.037)
	50	20.500 (±5.036)	0.000 (±0.000)	1.933 (±1.337)	90.533 (±66.951)	14.848 (±0.122)	16.685 (±0.163)
	75	24.133 (±6.163)	0.000 (±0.000)	1.800 (±1.243)	45.133 (±25.962)	15.927 (±0.126)	14.613 (±0.295)
	100	**33.033 (±10.791)**	**0.000 (±0.000)**	**1.767 (±1.251)**	**44.400 (±36.541)**	**16.267 (±0.136)**	**13.041 (±0.587)**

achieved when IS=699 and W1=100 which gave an average number of 33.033 ALCs in the active set of ALCs. An average number of 0.000 of the ALCs in the active set have memory status since there is only one iteration. The ALCs classified 44.400 patterns falsely as malignant and 1.767 as benign giving a misclassification of 46.167 patterns ($\#Misclassified = 44.400 + 1.767 = 46.167$) which gives a correct classification rate of 93.395%.

Table 7.10 shows the results when the ALCs were trained with the positive selection method. The best classification result was achieved when IS=175 and W1=75 which gave an average number of 22.308 ALCs in the active set of ALCs. An average number of 11.525 of the ALCs in the active set had memory status. The ALCs classified 40.333 patterns falsely as malignant and 1.967 as benign which gives a misclassification of 42.300 patterns ($\#Misclassified = 40.333 + 1.967 = 42.300$) and a correct classification rate of 93.948%. Figure 7.5 shows that the initial number of ALCs in the active set of ALCs was on average 26.566 and decreased over four iterations to an average of 15.333. Therefor the average fitness of the ALC set decreased over all iterations from 15.588 to 15.458. The average number of misclassification increased from iteration one to two and then decreased to iteration four.

7.2.3 Conclusion: Wisconsin Breast Cancer

In conclusion, the ALCs trained on the *benign* class with negative selection had a correct classification of 98.907% with an average number of 34.567 ALCs that had an average HD of 17.499. The ALCs that had been trained with positive selection on the *benign* class as self had a correct classification of 98.798% with an average number of 33.800 ALCs that had an average HD of 17.490. Comparing these results shows that there is neither a major difference in correct classification between the two different training methods nor in the average number of ALCs or HD with the same parameter settings (IS=524, W1=25), though negative selection does have a slightly better correct classification than positive selection. With the patterns from the *malignant* class as the self set the negative selection method had a correct classification of 93.395% with an average number of 33.033 ALCs that had an average HD of 13.041. The positive selection method had a correct classification of 93.948% with an average number of 22.308 ALCs that had an average HD of 14.322. These results conclude that when the patterns of the *malignant* class is used as the self set, different parameter settings are necessary to achieve similar correct classification results for both the selection methods and that the average number of ALCs for negative selection is higher than the average number of ALCs for positive selection. The difference in the average number of ALCs indicates that the patterns from the *benign* class have a

Table 7.10: *Malignant* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
175	25	22.333 (±4.240)	6.350 (±1.637)	2.767 (±1.695)	126.000 (±90.812)	22.037 (±0.151)	16.800 (±0.360)
	50	16.275 (±5.096)	6.375 (±2.677)	1.367 (±1.217)	106.167 (±76.795)	21.053 (±0.181)	15.923 (±0.420)
	75	**22.308 (±5.555)**	**11.525 (±3.397)**	**1.967 (±1.377)**	**40.333 (±19.752)**	**20.043 (±0.121)**	**14.322 (±0.267)**
	100	26.117 (±8.666)	13.933 (±5.283)	1.467 (±1.456)	62.733 (±75.871)	19.680 (±0.129)	12.643 (±0.633)
350	25	31.833 (±6.701)	7.950 (±2.291)	2.000 (±1.259)	126.500 (±99.243)	22.419 (±0.100)	17.741 (±0.043)
	50	21.100 (±4.708)	6.967 (±1.857)	2.000 (±1.390)	98.167 (±60.796)	21.190 (±0.136)	16.696 (±0.169)
	75	24.667 (±6.315)	9.733 (±2.693)	1.967 (±1.564)	60.800 (±47.323)	20.111 (±0.109)	14.685 (±0.309)
	100	28.867 (±13.577)	11.817 (±5.759)	2.067 (±1.530)	61.400 (±50.348)	19.798 (±0.169)	13.080 (±0.439)
524	25	32.500 (±4.524)	8.933 (±1.973)	1.933 (±1.437)	109.667 (±61.938)	22.407 (±0.097)	17.741 (±0.034)
	50	20.400 (±6.414)	7.300 (±2.753)	2.033 (±1.217)	115.767 (±86.717)	21.225 (±0.177)	16.698 (±0.148)
	75	25.700 (±6.944)	11.167 (±3.049)	1.967 (±0.928)	54.600 (±41.608)	20.026 (±0.104)	14.581 (±0.303)
	100	32.267 (±14.350)	14.200 (±6.504)	1.933 (±1.437)	55.100 (±61.477)	19.752 (±0.140)	12.931 (±0.530)
699	25	28.600 (±9.227)	0.000 (±0.000)	1.867 (±1.408)	164.033 (±105.704)	22.505 (±0.209)	17.723 (±0.049)
	50	19.800 (±5.054)	0.000 (±0.000)	2.000 (±1.114)	114.900 (±79.443)	21.182 (±0.106)	16.669 (±0.125)
	75	26.867 (±4.939)	0.000 (±0.000)	1.800 (±1.095)	45.800 (±21.815)	20.093 (±0.135)	14.733 (±0.294)
	100	29.333 (±10.908)	0.000 (±0.000)	1.667 (±1.241)	60.900 (±64.088)	19.712 (±0.143)	13.016 (±0.481)

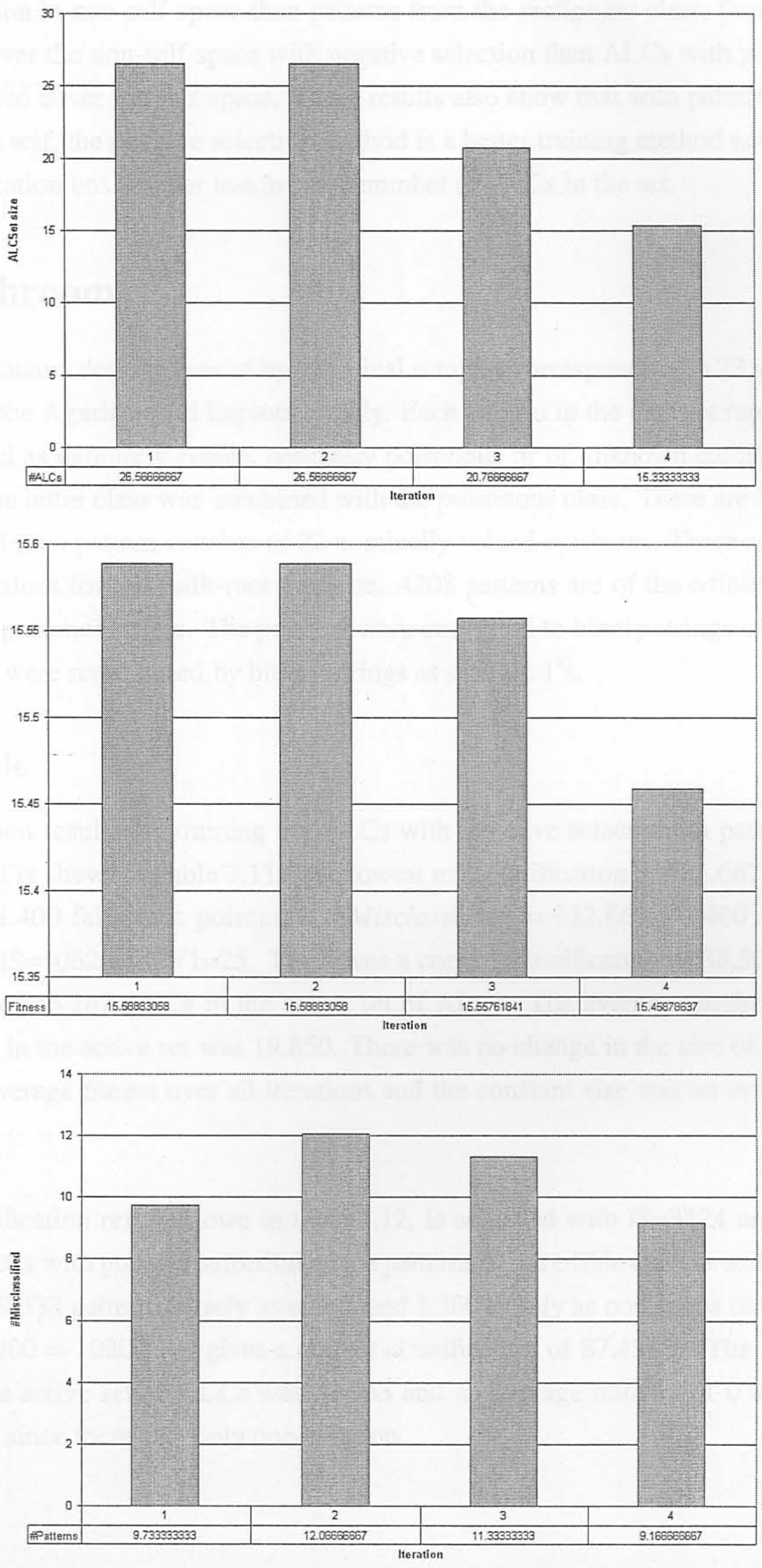


Figure 7.6: *Malignant* - Positive selection with IS=175 and W1=75

larger distribution in non-self space than patterns from the *malignant* class, thus more ALCs is necessary to cover the non-self space with negative selection than ALCs with positive selection that only needs to cover the self space. These results also show that with patterns from the *malignant* class as self, the positive selection method is a better training method not only for better correct classification but also for less average number of ALCs in the set.

7.3 Mushroom

The data set contains descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each pattern in the data set represents a specie that is classified as definitely edible, definitely poisonous or of unknown edibility and not recommended. The latter class was combined with the poisonous class. There are 8124 patterns in the data set and each pattern consists of 22 nominally valued attributes. There are 2480 patterns with missing values for the stalk-root attribute. 4208 patterns are of the edible class and 3916 patterns of the poisonous class. The patterns were converted to binary strings of length 57. The missing values were represented by binary strings as straight 1's.

7.3.1 Edible

The classification results for training the ALCs with negative selection on patterns of the edible class as self is shown in table 7.11. The lowest misclassification of 932.667 patterns falsely as edible and 1.400 falsely as poisonous ($\#Misclassified = 932.667 + 1.400 = 934.067$) was achieved with IS=4062 and W1=25. This gives a correct classification of 88.502% with an average number of 46.267 ALCs in the active set of ALCs. The average number of ALCs with memory status in the active set was 19.850. There was no change in the size of the active set of ALCs or the average fitness over all iterations and the constant size was an average number of 46.267 ALCs.

The best classification result shown in table 7.12, is achieved with IS=8124 and W1=25 when training the ALCs with positive selection on the patterns of the edible class as self. The misclassification of 1019.333 patterns falsely as edible and 1.200 falsely as poisonous ($\#Misclassified = 1019.333 + 1.200 = 1020.533$) gives a correct classification of 87.438%. The average number of ALCs in the active set of ALCs was 43.933 and an average number of 0.000 of these had memory status since there was only one iteration.

Table 7.11: *Edible* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
2031	25	37.925 (±4.890)	17.675 (±3.150)	0.967 (±1.351)	949.867 (±195.539)	25.682 (±0.183)	27.475 (±0.100)
	50	26.333 (±8.039)	13.625 (±4.860)	1.100 (±2.023)	1376.800 (±448.271)	28.627 (±0.148)	24.874 (±0.252)
	75	34.125 (±11.146)	19.583 (±7.072)	1.267 (±2.180)	1276.367 (±274.065)	30.695 (±0.232)	21.749 (±0.330)
	100	63.800 (±18.449)	39.900 (±12.374)	1.867 (±2.738)	1172.033 (±290.378)	31.466 (±0.193)	20.162 (±0.337)
4062	**25**	**46.267 (±3.999)**	**19.850 (±2.297)**	**1.400 (±2.222)**	**932.667 (±196.295)**	**25.520 (±0.131)**	**27.698 (±0.063)**
	50	28.067 (±8.358)	13.017 (±4.128)	1.167 (±1.704)	1451.367 (±449.082)	28.554 (±0.239)	25.009 (±0.222)
	75	43.833 (±11.641)	20.783 (±5.640)	1.700 (±2.996)	1224.100 (±359.480)	30.706 (±0.160)	21.845 (±0.361)
	100	74.733 (±18.221)	35.467 (±8.684)	2.767 (±4.368)	1083.733 (±186.907)	31.449 (±0.144)	20.224 (±0.251)
6093	25	45.033 (±4.390)	19.350 (±2.327)	2.000 (±3.373)	1021.633 (±216.756)	25.510 (±0.161)	27.704 (±0.051)
	50	26.767 (±8.097)	12.667 (±4.022)	1.067 (±1.552)	1300.867 (±314.306)	28.534 (±0.214)	25.041 (±0.204)
	75	45.800 (±9.568)	21.933 (±4.686)	1.467 (±2.417)	1221.533 (±253.926)	30.687 (±0.136)	21.879 (±0.249)
	100	63.733 (±17.416)	30.500 (±8.270)	2.667 (±3.994)	1171.700 (±202.250)	31.391 (±0.137)	20.303 (±0.296)
8124	25	45.333 (±4.011)	0.000 (±0.000)	1.533 (±1.943)	981.233 (±314.111)	25.498 (±0.154)	27.709 (±0.064)
	50	29.267 (±9.836)	0.000 (±0.000)	1.267 (±1.856)	1387.433 (±541.728)	28.609 (±0.276)	24.960 (±0.251)
	75	42.733 (±10.544)	0.000 (±0.000)	1.900 (±2.987)	1244.667 (±262.081)	30.701 (±0.213)	21.909 (±0.369)
	100	65.867 (±17.702)	0.000 (±0.000)	1.967 (±3.023)	1120.867 (±187.307)	31.451 (±0.132)	20.197 (±0.283)

Table 7.12: *Edible* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
2031	25	39.417 (±3.322)	18.500 (±2.310)	1.600 (±2.647)	1041.067 (±208.822)	31.348 (±0.167)	27.516 (±0.116)
	50	23.317 (±7.704)	11.758 (±4.421)	0.667 (±1.028)	1459.433 (±482.058)	28.467 (±0.209)	24.794 (±0.324)
	75	40.308 (±10.520)	23.492 (±7.175)	1.600 (±2.568)	1175.733 (±267.085)	26.284 (±0.164)	21.788 (±0.329)
	100	60.367 (±16.348)	37.025 (±11.319)	2.333 (±4.205)	1147.567 (±263.109)	25.501 (±0.204)	20.128 (±0.386)
4062	25	42.733 (±9.487)	17.717 (±4.211)	1.500 (±1.996)	1146.200 (±454.393)	31.546 (±0.217)	27.681 (±0.082)
	50	27.300 (±8.408)	12.633 (±3.792)	1.367 (±2.356)	1470.900 (±403.862)	28.449 (±0.218)	25.036 (±0.246)
	75	46.833 (±11.293)	22.333 (±5.284)	1.567 (±2.112)	1164.233 (±292.382)	26.275 (±0.176)	21.882 (±0.323)
	100	62.367 (±17.824)	29.683 (±8.405)	2.233 (±3.645)	1149.600 (±241.385)	25.535 (±0.169)	20.176 (±0.290)
6093	25	44.033 (±7.323)	19.450 (±3.539)	1.300 (±1.705)	1098.233 (±398.745)	31.574 (±0.260)	27.707 (±0.051)
	50	28.367 (±8.344)	13.350 (±4.067)	0.867 (±1.634)	1437.400 (±409.499)	28.416 (±0.225)	25.027 (±0.218)
	75	44.467 (±11.073)	21.367 (±5.597)	1.967 (±2.942)	1212.500 (±264.740)	26.388 (±0.214)	22.012 (±0.394)
	100	68.067 (±18.431)	32.583 (±8.747)	3.133 (±4.547)	1205.900 (±229.452)	25.643 (±0.176)	20.350 (±0.327)
8124	**25**	**43.933 (±7.506)**	**0.000 (±0.000)**	**1.200 (±1.495)**	**1019.333 (±373.054)**	**31.502 (±0.153)**	**27.690 (±0.059)**
	50	29.533 (±8.464)	0.000 (±0.000)	1.100 (±2.040)	1322.200 (±332.097)	28.340 (±0.202)	24.925 (±0.238)
	75	39.200 (±10.584)	0.000 (±0.000)	1.800 (±2.809)	1250.800 (±308.670)	26.322 (±0.187)	21.865 (±0.361)
	100	71.133 (±20.669)	0.000 (±0.000)	2.167 (±3.064)	1205.633 (±283.299)	25.535 (±0.158)	20.159 (±0.339)

7.3.2 Poisonous

The results for training the ALCs with the negative selection method on patterns of the poisonous class as self is shown in table 7.13. The best classification result was achieved when IS=8124 and W1=25 which gave an average number of 45.033 ALCs in the active set of ALCs. An average number of 0.000 of the ALCs in the active set had memory status since there is only one iteration. The ALCs classified 1591.933 patterns falsely as poisonous and 1.700 as edible giving a misclassification of 1593.633 patterns ($\#Misclassified = 1591.933 + 1.700 = 1593.633$) which gives a correct classification rate of 80.383%.

The best classification result shown in table 7.14, was achieved with IS=2031 and W1=100 when training the ALCs with positive selection on the patterns of the poisonous class as self. The misclassification of 1695.000 patterns falsely as poisonous and 2.467 falsely as edible patterns ($\#Misclassified = 1695.000 + 2.467 = 1697.467$) gives a correct classification of 79.105%. The average number of ALCs in the active set of ALCs was 50.717 and an average number of 17.242 of these had memory status. Figure 7.6 shows that the initial number of ALCs in the active set of ALCs was on average 64.033 and decreased over four iterations to an average of 26.166, but different from the previous experiments the average fitness of the ALC set increased over four iterations from 31.278 to 31.429. The average number of misclassification increased from iteration one to two and then decreased to iteration four.

7.3.3 Conclusion: Mushroom

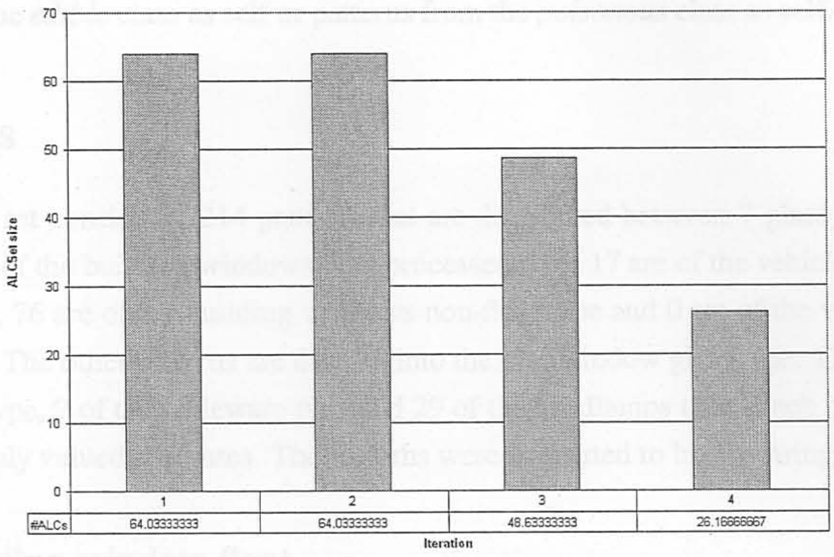
Comparing the different selection methods with patterns from the *edible* class as the self set, it can be concluded that the negative selection method has a slightly better correct classification than the positive selection method. Both the selection methods have a value of 25 for W1 but different values for IS to obtain the best classification results. The same value of 25 for W1 indicates that the fitness of the ALCs in the GA for both selection methods is more influenced by their HD than by their ADT. The HD of both selection methods differ with 0.008 and indicates that the amount of overlap is more or less the same for both selection methods. When patterns from the *poisonous* class is used as the self set then the negative selection method has a better classification performance than the positive selection method since not only does the negative selection method have better correct classification results but also has on average less ALCs than the positive selection method to classify the patterns. These results indicate that the negative selection method has better classification performance than the positive selection method with

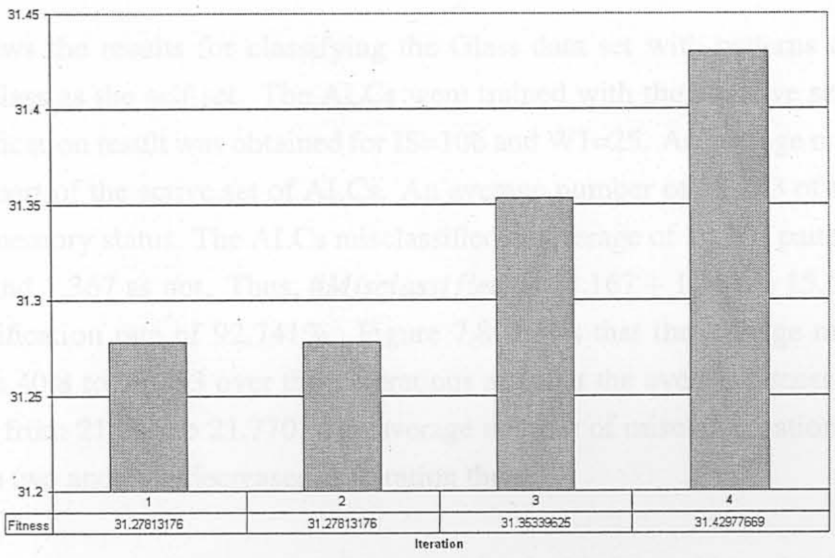
Table 7.13: *Poisonous* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
2031	25	34.925 (±5.412)	10.383 (±2.183)	2.100 (±2.928)	1876.300 (±529.147)	25.609 (±0.222)	27.371 (±0.191)
	50	23.858 (±5.654)	7.158 (±2.091)	1.833 (±1.949)	2239.167 (±530.783)	28.477 (±0.266)	24.864 (±0.365)
	75	33.575 (±10.097)	11.217 (±3.987)	2.033 (±3.243)	1917.000 (±458.657)	30.609 (±0.199)	21.728 (±0.389)
	100	52.792 (±14.391)	18.042 (±5.996)	3.033 (±4.687)	1833.933 (±425.009)	31.269 (±0.158)	20.342 (±0.285)
4062	25	46.333 (±2.551)	17.217 (±1.804)	1.400 (±1.754)	1834.900 (±382.288)	25.535 (±0.113)	27.722 (±0.048)
	50	28.533 (±8.245)	11.350 (±3.462)	1.200 (±1.375)	2254.333 (±527.809)	28.446 (±0.209)	25.154 (±0.184)
	75	44.300 (±11.928)	17.167 (±4.652)	1.867 (±2.460)	1825.600 (±342.097)	30.529 (±0.156)	22.074 (±0.263)
	100	66.400 (±21.888)	24.617 (±8.242)	2.833 (±3.505)	1686.100 (±474.694)	31.288 (±0.197)	20.340 (±0.378)
6093	25	44.700 (±7.498)	18.967 (±3.704)	1.767 (±1.478)	1823.233 (±527.523)	25.530 (±0.187)	27.718 (±0.059)
	50	29.933 (±9.025)	13.517 (±4.419)	1.300 (±2.020)	2236.967 (±462.290)	28.499 (±0.210)	25.094 (±0.282)
	75	40.800 (±10.610)	18.183 (±4.700)	1.433 (±1.813)	1952.600 (±474.423)	30.487 (±0.139)	22.035 (±0.308)
	100	63.867 (±17.338)	28.167 (±7.863)	2.467 (±2.956)	1746.633 (±462.588)	31.280 (±0.120)	20.370 (±0.224)
8124	**25**	**45.033 (±4.582)**	**0.000 (±0.000)**	**1.700 (±2.003)**	**1591.933 (±382.550)**	**25.523 (±0.142)**	**27.729 (±0.049)**
	50	27.567 (±8.736)	0.000 (±0.000)	1.367 (±1.608)	2218.267 (±498.915)	28.509 (±0.218)	25.052 (±0.235)
	75	42.167 (±13.099)	0.000 (±0.000)	1.533 (±1.925)	1889.133 (±391.633)	30.488 (±0.152)	22.089 (±0.289)
	100	57.633 (±24.102)	0.000 (±0.000)	1.867 (±2.460)	1940.833 (±569.112)	31.267 (±0.228)	20.275 (±0.522)

Table 7.14: *Poisonous* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
2031	25	36.433 (±3.570)	11.075 (±1.670)	2.033 (±2.580)	1748.767 (±335.773)	31.338 (±0.188)	27.385 (±0.114)
	50	25.033 (±6.828)	7.850 (±2.505)	1.567 (±1.888)	2095.600 (±464.683)	28.493 (±0.265)	24.844 (±0.284)
	75	32.792 (±9.468)	10.558 (±3.776)	1.500 (±1.796)	1989.933 (±504.108)	26.487 (±0.214)	21.843 (±0.372)
	100	**50.717 (±13.291)**	**17.242 (±5.158)**	**2.467 (±3.391)**	**1695.000 (±401.959)**	**25.665 (±0.178)**	**20.201 (±0.336)**
4062	25	43.867 (±8.025)	15.967 (±3.181)	1.700 (±2.395)	1813.733 (±587.445)	31.556 (±0.340)	27.737 (±0.067)
	50	28.333 (±9.060)	11.117 (±4.023)	0.867 (±1.306)	2281.133 (±449.601)	28.611 (±0.219)	25.177 (±0.212)
	75	41.200 (±9.477)	15.867 (±4.013)	1.867 (±2.330)	1949.633 (±311.235)	26.460 (±0.177)	22.024 (±0.287)
	100	59.900 (±17.574)	22.067 (±6.700)	2.600 (±3.058)	1773.900 (±491.493)	25.779 (±0.162)	20.377 (±0.287)
6093	25	45.400 (±6.946)	19.517 (±3.158)	1.933 (±2.067)	1712.333 (±479.901)	31.506 (±0.246)	27.727 (±0.054)
	50	27.100 (±8.763)	12.100 (±4.229)	1.233 (±2.096)	2221.867 (±541.412)	28.526 (±0.235)	25.096 (±0.252)
	75	39.667 (±11.133)	17.783 (±5.051)	1.700 (±1.784)	2069.133 (±343.100)	26.489 (±0.188)	21.993 (±0.394)
	100	68.233 (±21.837)	30.267 (±9.868)	2.467 (±3.137)	1704.767 (±562.829)	25.748 (±0.198)	20.400 (±0.382)
8124	25	45.733 (±3.741)	0.000 (±0.000)	1.967 (±2.632)	1843.533 (±334.025)	31.480 (±0.125)	27.725 (±0.049)
	50	31.700 (±7.996)	0.000 (±0.000)	1.500 (±2.662)	2111.500 (±360.474)	28.498 (±0.178)	25.163 (±0.213)
	75	39.867 (±11.365)	0.000 (±0.000)	1.533 (±2.097)	1886.267 (±496.066)	26.526 (±0.172)	22.101 (±0.386)
	100	68.100 (±18.054)	0.000 (±0.000)	3.100 (±3.595)	1729.967 (±474.410)	25.787 (±0.198)	20.445 (±0.302)





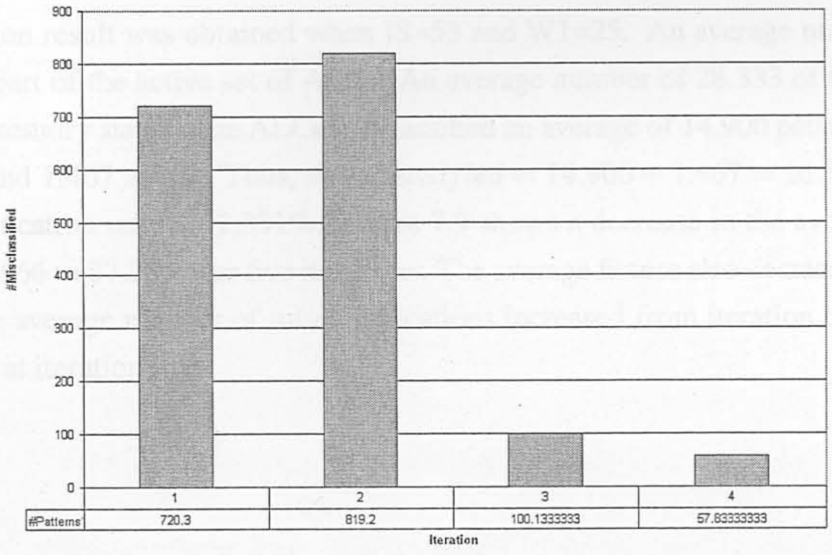


Figure 7.7: *Poisonous* - Positive selection with IS=2031 and W1=100

patterns from the *edible* class as self or patterns from the *poisonous* class as self.

7.4 Glass

The glass data set consists of 214 patterns that are distributed between 7 glass types (classes). 70 patterns are of the building windows float processed type, 17 are of the vehicle windows float processed type, 76 are of the building windows non-float type and 0 are of the vehicle windows non-float type. The other patterns are divided into the non-window glass type: 13 patterns are of the container type, 9 of the tableware type and 29 of the headlamps type. Each patterns consists of 9 continuously valued attributes. The patterns were converted to binary strings of length 45.

7.4.1 Building-window-float

Table 7.15 shows the results for classifying the Glass data set with patterns of the building-window-float class as the self set. The ALCs were trained with the negative selection method. The best classification result was obtained for IS=106 and W1=25. An average number of 40.444 ALCs formed part of the active set of ALCs. An average number of 25.233 of the ALCs in the active set had memory status. The ALCs misclassified an average of 14.167 patterns as building-window-float and 1.367 as not. Thus, $\#Misclassified = 14.167 + 1.367 = 15.534$ which gives a correct classification rate of 92.741%. Figure 7.8 shows that the average number of ALCs decreased from 40.8 to 39.733 over three iterations and that the average fitness of the ALC set also decreased from 21.796 to 21.770. The average number of misclassification increased from iteration one to two and then decreased at iteration three.

Table 7.16 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=53 and W1=25. An average number of 39.167 ALCs formed part of the active set of ALCs. An average number of 28.333 of the ALCs in the active set had memory status. The ALCs misclassified an average of 14.900 patterns as building-window-float and 1.467 as not. Thus, $\#Misclassified = 14.900 + 1.467 = 16.367$ which gives a correct classification rate of 92.351%. Figure 7.9 shows a decrease in the average number of ALCs from 40.666 to 37.266 over five iterations. The average fitness also decreased from 21.790 to 21.722. The average number of misclassifications increased from iteration one to three and then decreased at iteration five.

Table 7.15: *Building-window-float* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	37.913 (±4.368)	27.353 (±3.741)	1.233 (±2.144)	14.967 (±5.129)	21.977 (±0.122)	21.690 (±0.065)
	50	23.467 (±5.185)	16.600 (±4.251)	1.033 (±1.810)	28.533 (±9.634)	24.698 (±0.210)	19.175 (±0.272)
	75	36.100 (±9.191)	27.433 (±7.420)	1.367 (±1.608)	26.567 (±6.956)	26.454 (±0.196)	16.549 (±0.256)
	100	56.100 (±13.547)	43.087 (±10.538)	1.133 (±1.871)	23.700 (±7.720)	26.953 (±0.169)	15.319 (±0.329)
106	**25**	**40.444 (±4.020)**	**25.233 (±2.644)**	**1.367 (±1.974)**	**14.167 (±4.662)**	**21.946 (±0.146)**	**21.735 (±0.051)**
	50	25.767 (±6.106)	16.289 (±4.260)	1.200 (±1.472)	27.000 (±7.883)	24.675 (±0.169)	19.229 (±0.180)
	75	35.933 (±8.158)	23.189 (±5.389)	1.400 (±1.773)	25.733 (±8.718)	26.424 (±0.184)	16.597 (±0.306)
	100	53.400 (±12.041)	34.856 (±8.013)	1.200 (±1.846)	23.233 (±6.095)	26.925 (±0.190)	15.334 (±0.261)
160	25	40.100 (±4.318)	19.500 (±2.125)	1.433 (±1.960)	15.467 (±4.939)	21.857 (±0.141)	21.762 (±0.052)
	50	25.667 (±5.996)	12.783 (±3.019)	1.167 (±2.019)	26.867 (±10.126)	24.702 (±0.237)	19.198 (±0.182)
	75	35.300 (±9.855)	17.633 (±4.936)	1.067 (±1.660)	28.367 (±7.522)	26.436 (±0.227)	16.521 (±0.326)
	100	56.267 (±19.490)	28.117 (±9.749)	1.133 (±1.634)	23.967 (±8.736)	26.986 (±0.226)	15.280 (±0.359)
214	25	39.667 (±2.832)	0.000 (±0.000)	1.367 (±1.991)	16.800 (±4.759)	21.841 (±0.185)	21.778 (±0.071)
	50	24.467 (±7.171)	0.000 (±0.000)	1.067 (±1.639)	28.800 (±8.290)	24.702 (±0.247)	19.195 (±0.218)
	75	30.700 (±6.914)	0.000 (±0.000)	1.167 (±1.840)	28.967 (±8.977)	26.444 (±0.219)	16.493 (±0.282)
	100	54.533 (±14.498)	0.000 (±0.000)	1.367 (±1.629)	24.800 (±6.150)	26.945 (±0.246)	15.332 (±0.380)

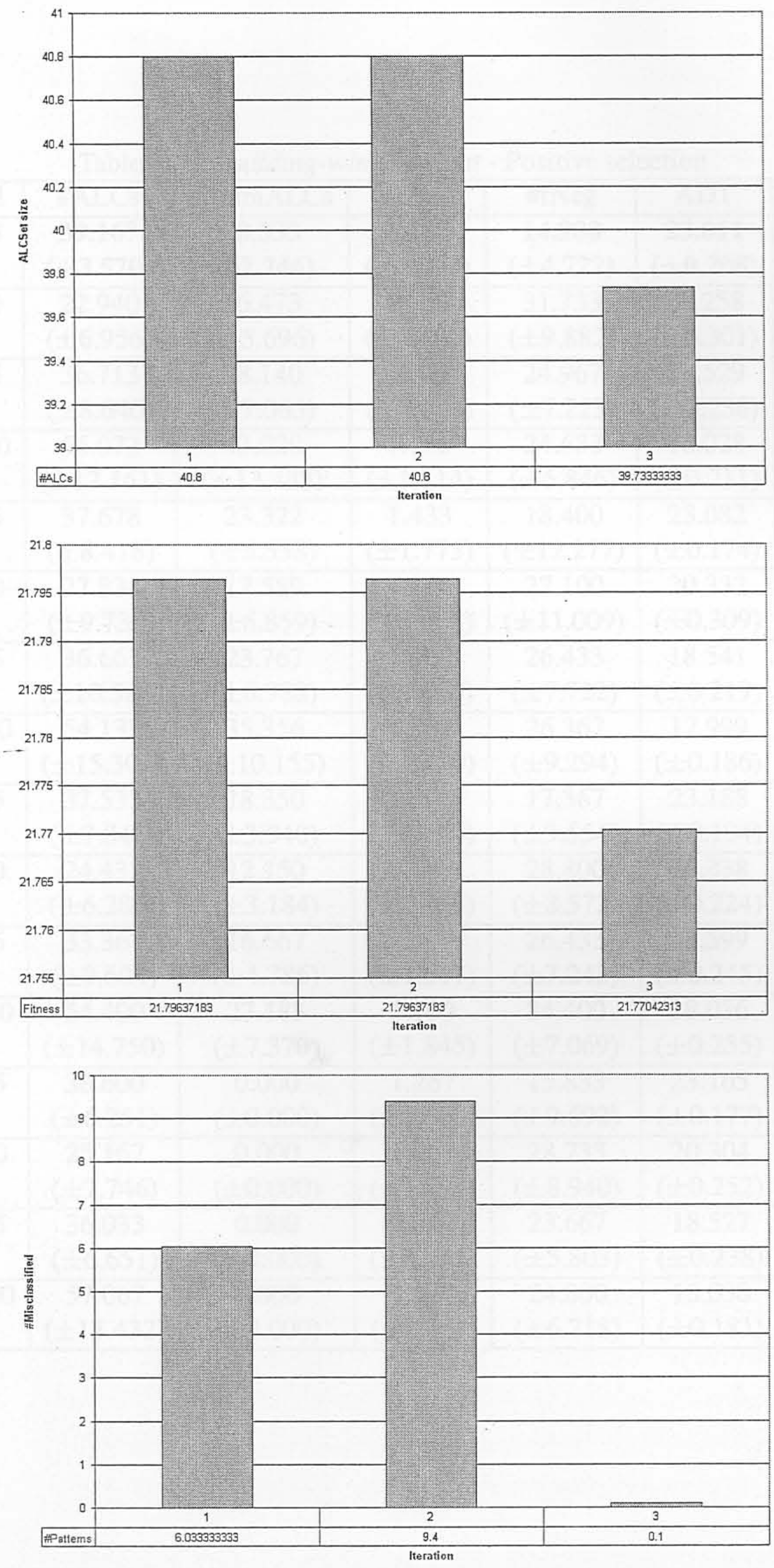


Figure 7.8: *Building-window-float* - Negative selection with IS=106 and W1=25

Table 7.16: *Building-window-float* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	**25**	**39.167 (±3.570)**	**28.333 (±2.746)**	**1.467 (±1.978)**	**14.900 (±4.722)**	**23.011 (±0.208)**	**21.681 (±0.099)**
	50	22.940 (±6.956)	16.473 (±5.696)	1.133 (±1.479)	31.733 (±9.882)	20.258 (±0.301)	19.088 (±0.308)
	75	36.713 (±8.640)	28.140 (±7.063)	1.033 (±2.008)	24.967 (±7.223)	18.529 (±0.236)	16.533 (±0.363)
	100	56.073 (±17.161)	43.020 (±13.433)	1.133 (±1.814)	24.633 (±6.846)	18.028 (±0.211)	15.330 (±0.219)
106	25	37.678 (±8.418)	23.322 (±5.558)	1.433 (±1.775)	18.400 (±12.277)	23.082 (±0.174)	21.691 (±0.187)
	50	27.833 (±9.739)	17.589 (±6.859)	1.100 (±1.863)	27.100 (±11.009)	20.332 (±0.309)	19.225 (±0.217)
	75	36.667 (±10.584)	23.767 (±6.988)	1.100 (±1.668)	26.433 (±7.722)	18.541 (±0.219)	16.511 (±0.270)
	100	54.133 (±15.301)	35.356 (±10.155)	1.367 (±1.790)	26.367 (±9.294)	17.999 (±0.186)	15.271 (±0.290)
160	25	37.533 (±7.847)	18.350 (±3.940)	1.300 (±1.664)	17.367 (±9.554)	23.188 (±0.194)	21.772 (±0.063)
	50	24.433 (±6.285)	12.150 (±3.184)	1.067 (±1.856)	28.800 (±8.572)	20.338 (±0.224)	19.214 (±0.190)
	75	33.367 (±9.604)	16.667 (±4.786)	0.900 (±1.517)	26.433 (±7.243)	18.599 (±0.245)	16.624 (±0.369)
	100	54.400 (±14.750)	27.183 (±7.370)	1.100 (±1.845)	24.400 (±7.069)	18.036 (±0.235)	15.338 (±0.278)
214	25	38.600 (±6.251)	0.000 (±0.000)	1.267 (±1.660)	15.833 (±9.692)	23.165 (±0.177)	21.754 (±0.056)
	50	25.167 (±7.746)	0.000 (±0.000)	1.067 (±1.799)	28.733 (±8.940)	20.304 (±0.252)	19.240 (±0.213)
	75	36.033 (±6.651)	0.000 (±0.000)	1.367 (±2.141)	23.667 (±5.803)	18.527 (±0.238)	16.512 (±0.246)
	100	57.067 (±15.432)	0.000 (±0.000)	1.167 (±2.001)	24.800 (±6.718)	18.038 (±0.181)	15.369 (±0.269)

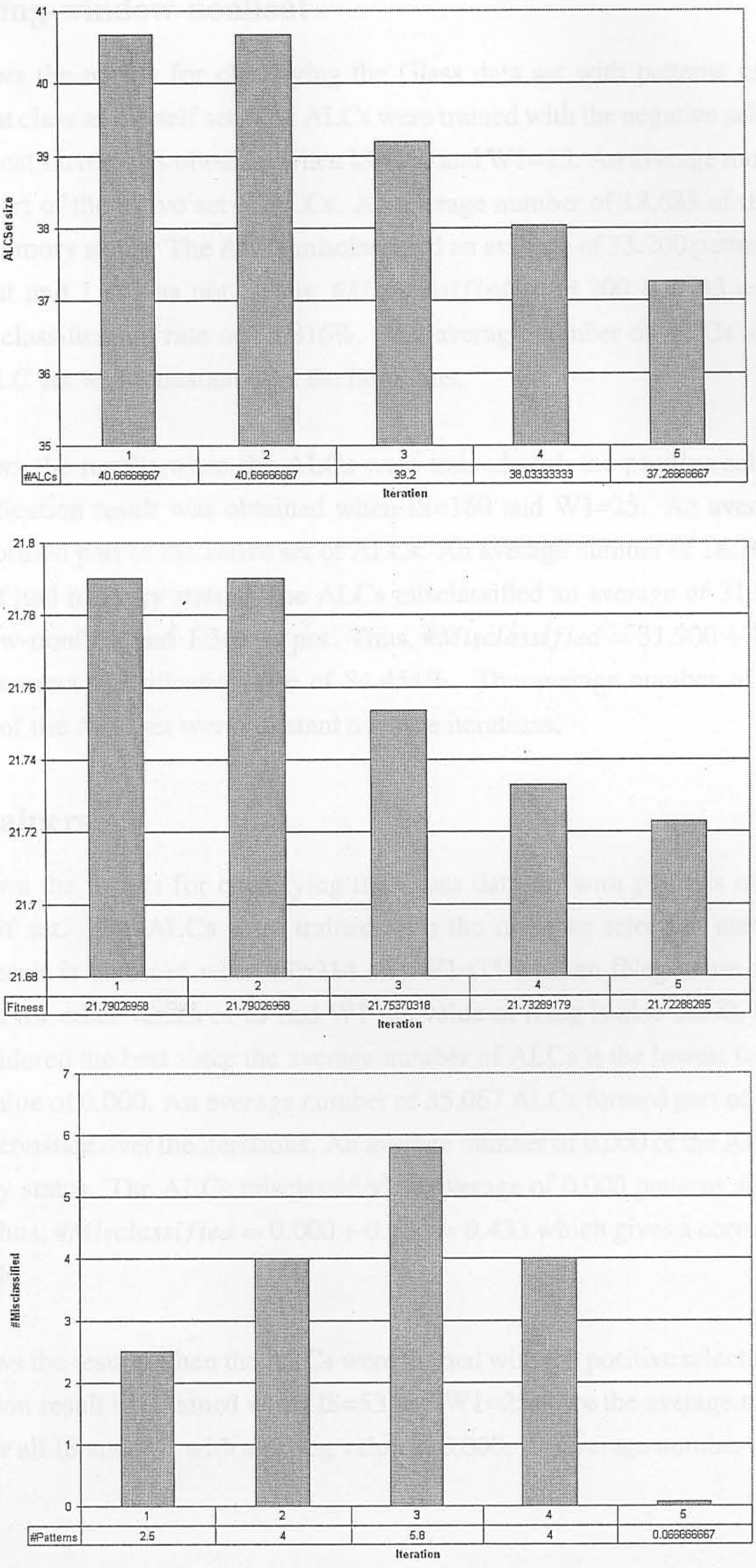


Figure 7.9: *Building-window-float* - Positive selection with IS=53 and W1=25

7.4.2 Building-window-nonfloat

Table 7.17 shows the results for classifying the Glass data set with patterns of the building-window-nonfloat class as the self set. The ALCs were trained with the negative selection method. The best classification result is obtained when IS=160 and W1=25. An average number of 39.700 ALCs formed part of the active set of ALCs. An average number of 18.683 of the ALCs in the active set had memory status. The ALCs misclassified an average of 33.200 patterns as building-window-nonfloat and 1.433 as not. Thus, $\#Misclassified = 33.200 + 1.433 = 34.633$ which gives a correct classification rate of 83.816%. The average number of ALCs and the average fitness of the ALC set were constant over the iterations.

Table 7.18 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=160 and W1=25. An average number of 39.800 ALCs formed part of the active set of ALCs. An average number of 18.700 of the ALCs in the active set had memory status. The ALCs misclassified an average of 31.900 patterns as building-window-nonfloat and 1.367 as not. Thus, $\#Misclassified = 31.900 + 1.367 = 33.267$ which gives a correct classification rate of 84.454%. The average number of ALCs and the average fitness of the ALC set were constant over the iterations.

7.4.3 Containers

Table 7.19 shows the results for classifying the Glass data set with patterns of the containers class as the self set. The ALCs were trained with the negative selection method. The best classification result is obtained when IS=214 and W1=75 with an fNeg value of 0.000. Table 7.19 shows that for other values of IS and W1 the value of fNeg is also 0.000, but IS=214 and W1=75 is considered the best since the average number of ALCs is the lowest for all IS and W1 with an fNeg value of 0.000. An average number of 35.067 ALCs formed part of the active set of ALCs and was constant over the iterations. An average number of 0.000 of the ALCs in the active set had memory status. The ALCs misclassified an average of 0.000 patterns as containers and 0.433 as not. Thus, $\#Misclassified = 0.000 + 0.433 = 0.433$ which gives a correct classification rate of 99.797%.

Table 7.20 shows the results when the ALCs were trained with the positive selection method. The best classification result is obtained when IS=53 and W1=25 since the average number of ALCs is the lowest for all IS and W1 with an fNeg value of 0.000. An average number of 38.013 ALCs

Table 7.17: *Building-window-nonfloat* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	36.540 (±3.030)	23.720 (±2.456)	1.400 (±1.976)	33.300 (±6.115)	21.246 (±0.192)	21.737 (±0.109)
	50	22.947 (±7.384)	15.593 (±5.863)	1.167 (±2.069)	53.300 (±13.916)	23.661 (±0.219)	19.449 (±0.264)
	75	33.927 (±9.090)	23.587 (±6.656)	1.367 (±1.991)	49.733 (±11.104)	25.373 (±0.211)	16.817 (±0.352)
	100	40.500 (±13.097)	27.660 (±9.356)	1.433 (±2.885)	53.467 (±12.610)	25.872 (±0.236)	15.515 (±0.331)
106	25	38.322 (±3.355)	22.178 (±2.494)	1.433 (±2.176)	34.900 (±5.851)	21.159 (±0.119)	21.813 (±0.047)
	50	25.211 (±6.388)	15.000 (±4.245)	1.200 (±2.340)	49.233 (±10.757)	23.680 (±0.260)	19.481 (±0.228)
	75	34.389 (±8.543)	21.211 (±5.974)	1.433 (±2.501)	51.933 (±12.051)	25.317 (±0.197)	16.910 (±0.280)
	100	46.400 (±17.230)	29.433 (±11.321)	1.433 (±2.515)	50.900 (±15.615)	25.858 (±0.202)	15.535 (±0.423)
160	**25**	**39.700 (±3.064)**	**18.683 (±1.774)**	**1.433 (±1.794)**	**33.200 (±7.889)**	**21.125 (±0.134)**	**21.836 (±0.048)**
	50	23.333 (±6.723)	11.317 (±3.480)	1.000 (±1.682)	52.367 (±12.417)	23.593 (±0.232)	19.566 (±0.209)
	75	32.800 (±9.722)	16.083 (±4.657)	1.267 (±1.837)	53.333 (±9.349)	25.280 (±0.202)	16.939 (±0.273)
	100	51.200 (±13.793)	25.183 (±6.801)	1.400 (±2.343)	47.800 (±9.729)	25.905 (±0.174)	15.534 (±0.268)
214	25	40.833 (±3.573)	0.000 (±0.000)	1.667 (±2.073)	33.733 (±7.027)	21.124 (±0.148)	21.844 (±0.051)
	50	24.933 (±6.214)	0.000 (±0.000)	1.233 (±1.995)	49.933 (±12.261)	23.640 (±0.244)	19.580 (±0.187)
	75	36.900 (±10.387)	0.000 (±0.000)	1.233 (±2.161)	50.133 (±10.837)	25.378 (±0.181)	16.827 (±0.252)
	100	50.533 (±17.156)	0.000 (±0.000)	1.167 (±2.198)	47.300 (±10.844)	25.861 (±0.206)	15.571 (±0.269)

Table 7.18: *Building-window-nonfloat* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	34.527 (±6.853)	22.193 (±5.086)	1.567 (±2.473)	35.800 (±12.510)	23.756 (±0.161)	21.691 (±0.196)
	50	24.107 (±6.210)	16.087 (±4.954)	1.200 (±1.669)	50.900 (±8.747)	21.293 (±0.172)	19.443 (±0.203)
	75	32.627 (±9.969)	22.553 (±6.964)	1.200 (±1.669)	51.333 (±10.896)	19.685 (±0.236)	16.858 (±0.284)
	100	44.360 (±16.150)	30.287 (±11.375)	1.233 (±2.528)	51.167 (±13.052)	19.056 (±0.199)	15.376 (±0.524)
106	25	38.889 (±4.168)	22.344 (±2.909)	1.567 (±2.674)	32.267 (±8.103)	23.827 (±0.166)	21.807 (±0.056)
	50	25.378 (±8.316)	15.311 (±5.013)	1.300 (±2.003)	50.300 (±12.103)	21.366 (±0.221)	19.552 (±0.188)
	75	36.056 (±10.013)	22.333 (±6.701)	1.200 (±1.690)	48.600 (±10.595)	19.634 (±0.198)	16.856 (±0.326)
	100	51.867 (±15.567)	33.044 (±10.076)	1.367 (±2.327)	48.533 (±11.355)	19.097 (±0.206)	15.520 (±0.354)
160	**25**	**39.800 (±3.671)**	**18.700 (±1.720)**	**1.367 (±2.157)**	**31.900 (±7.303)**	**23.879 (±0.119)**	**21.846 (±0.052)**
	50	25.567 (±7.035)	12.517 (±3.475)	1.167 (±2.379)	50.400 (±9.357)	21.338 (±0.270)	19.556 (±0.264)
	75	33.733 (±8.267)	16.567 (±3.939)	1.233 (±2.192)	49.367 (±10.294)	19.630 (±0.211)	16.844 (±0.312)
	100	49.067 (±13.115)	24.050 (±6.383)	1.233 (±2.029)	48.500 (±7.305)	19.158 (±0.227)	15.593 (±0.320)
214	25	39.100 (±3.854)	0.000 (±0.000)	1.467 (±2.515)	32.733 (±8.550)	23.905 (±0.163)	21.853 (±0.044)
	50	24.333 (±7.712)	0.000 (±0.000)	1.267 (±2.333)	53.500 (±12.500)	21.399 (±0.263)	19.561 (±0.176)
	75	34.000 (±10.567)	0.000 (±0.000)	1.167 (±2.534)	51.233 (±12.182)	19.626 (±0.210)	16.807 (±0.323)
	100	54.700 (±15.132)	0.000 (±0.000)	1.600 (±2.634)	46.500 (±7.615)	19.105 (±0.164)	15.595 (±0.267)

Table 7.19: *Containers* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	39.513 (±4.375)	29.387 (±6.604)	0.433 (±2.373)	0.000 (±0.000)	23.838 (±3.999)	21.753 (±0.152)
	50	29.800 (±8.195)	22.640 (±7.813)	0.433 (±2.373)	0.100 (±0.403)	26.582 (±3.483)	19.220 (±0.661)
	75	35.153 (±11.439)	27.773 (±9.901)	0.433 (±2.373)	0.367 (±0.928)	28.298 (±3.160)	16.435 (±1.143)
	100	55.520 (±19.997)	44.227 (±16.371)	0.433 (±2.373)	0.100 (±0.403)	28.874 (±3.051)	14.952 (±0.817)
106	25	39.567 (±4.034)	24.644 (±5.420)	0.433 (±2.373)	0.000 (±0.000)	23.780 (±4.010)	21.792 (±0.145)
	50	25.189 (±7.312)	15.922 (±5.727)	0.433 (±2.373)	0.067 (±0.254)	26.503 (±3.502)	19.260 (±0.685)
	75	34.400 (±9.565)	22.600 (±7.257)	0.433 (±2.373)	0.133 (±0.571)	28.339 (±3.151)	16.360 (±1.197)
	100	56.767 (±18.335)	37.722 (±12.550)	0.433 (±2.373)	0.000 (±0.000)	28.876 (±3.049)	14.917 (±0.517)
160	25	40.333 (±2.869)	19.167 (±3.905)	0.433 (±2.373)	0.000 (±0.000)	23.758 (±4.014)	21.822 (±0.136)
	50	24.167 (±6.711)	11.517 (±3.890)	0.433 (±2.373)	0.200 (±0.761)	26.518 (±3.500)	19.235 (±0.666)
	75	37.433 (±10.020)	18.533 (±5.578)	0.433 (±2.373)	0.167 (±0.461)	28.307 (±3.158)	16.456 (±1.140)
	100	63.433 (±22.201)	31.633 (±11.334)	0.433 (±2.373)	0.200 (±0.805)	28.917 (±3.042)	14.916 (±0.628)
214	25	40.900 (±3.537)	0.000 (±0.000)	0.433 (±2.373)	0.000 (±0.000)	23.796 (±4.007)	21.799 (±0.143)
	50	27.200 (±5.945)	0.000 (±0.000)	0.433 (±2.373)	0.167 (±0.592)	26.536 (±3.491)	19.280 (±0.627)
	75	**35.067 (±10.392)**	**0.000 (±0.000)**	**0.433 (±2.373)**	**0.000 (±0.000)**	**28.335 (±3.152)**	**16.437 (±1.157)**
	100	53.933 (±18.040)	0.000 (±0.000)	0.433 (±2.373)	0.000 (±0.000)	28.909 (±3.042)	14.943 (±0.808)

Table 7.20: *Containers* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	**25**	**38.013 (±6.682)**	**29.113 (±6.343)**	**0.433 (±2.373)**	**0.000 (±0.000)**	**21.125 (±3.992)**	**21.152 (±3.146)**
	50	26.620 (±9.261)	20.813 (±7.905)	0.433 (±2.373)	0.067 (±0.254)	18.496 (±3.502)	18.707 (±2.693)
	75	36.447 (±10.207)	29.013 (±8.476)	0.433 (±2.373)	0.033 (±0.183)	16.731 (±3.165)	15.915 (±2.193)
	100	55.320 (±22.065)	44.133 (±17.802)	0.433 (±2.373)	0.333 (±1.493)	16.132 (±3.053)	14.435 (±2.070)
106	25	38.622 (±5.288)	24.644 (±5.041)	0.433 (±2.373)	0.000 (±0.000)	21.187 (±4.003)	21.292 (±2.606)
	50	24.678 (±8.022)	15.989 (±5.940)	0.433 (±2.373)	0.167 (±0.531)	18.503 (±3.500)	18.801 (±2.152)
	75	36.311 (±13.129)	24.044 (±9.009)	0.433 (±2.373)	0.433 (±1.695)	16.689 (±3.159)	15.922 (±1.637)
	100	57.678 (±15.944)	38.400 (±10.817)	0.433 (±2.373)	0.000 (±0.000)	16.064 (±3.036)	14.492 (±1.615)
160	25	37.583 (±8.404)	18.267 (±5.273)	0.433 (±2.373)	0.200 (±0.925)	21.281 (±4.025)	21.418 (±1.923)
	50	24.633 (±6.294)	12.033 (±3.815)	0.433 (±2.373)	0.200 (±0.610)	18.492 (±3.499)	18.897 (±1.469)
	75	34.767 (±12.555)	17.283 (±6.533)	0.433 (±2.373)	0.300 (±0.952)	16.709 (±3.163)	16.042 (±0.981)
	100	50.133 (±17.459)	25.017 (±8.872)	0.433 (±2.373)	0.033 (±0.183)	16.029 (±3.030)	14.439 (±1.114)
214	25	39.767 (±3.692)	0.000 (±0.000)	0.433 (±2.373)	0.000 (±0.000)	21.233 (±4.014)	21.801 (±0.142)
	50	26.233 (±6.806)	0.000 (±0.000)	0.433 (±2.373)	0.333 (±1.155)	18.527 (±3.504)	19.334 (±0.630)
	75	33.100 (±9.517)	0.000 (±0.000)	0.433 (±2.373)	0.233 (±0.568)	16.721 (±3.164)	16.452 (±1.157)
	100	52.867 (±19.158)	0.000 (±0.000)	0.433 (±2.373)	0.167 (±0.747)	16.106 (±3.046)	14.900 (±0.841)

formed part of the active set of ALCs. An average number of 29.113 of the ALCs in the active set had memory status. The ALCs misclassified an average of 0.000 patterns as containers and 0.433 as not. Thus, $\#Misclassified = 0.000 + 0.433 = 0.433$ which gives a correct classification rate of 99.797%. Figure 7.10 shows that the average number of ALCs decreased from 39.866 to 36.733 over five iterations with a decreasing average fitness of the ALC set. The average number of misclassification decreased from iteration one to two, then increased again after iteration two and decreased at iteration five.

7.4.4 Headlamps

Table 7.21 shows the results for classifying the Glass data set with patterns of the headlamps class as the self set. The ALCs were trained with the negative selection method. The best classification result was obtained when IS=53 and W1=25 since the average number of ALCs was the lowest for all IS and W1 with an fNeg value of 0.367. An average number of 37.693 ALCs formed part of the active set of ALCs. An average number of 27.133 of the ALCs in the active set had memory status. The ALCs misclassified an average of 0.367 patterns as headlamps and 0.967 as not. Thus, $\#Misclassified = 0.367 + 0.967 = 1.334$ which gives a correct classification rate of 99.376%. Figure 7.11 shows a decreasing average number of ALCs, from 39.166 to 35.833 and a decreasing average fitness of the ALC set from 22.121 to 22.054 over five iterations. The average number of misclassification increased from iteration one to four and drops to 0.0 in iteration five.

Table 7.22 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=214 and W1=25. An average number of 39.000 ALCs formed part of the active set of ALCs and was constant over the iterations. An average number of 0.000 of the ALCs in the active set had memory status. The ALCs misclassified an average of 0.467 patterns as headlamps and 0.967 as not. Thus, $\#Misclassified = 0.467 + 0.967 = 1.434$ which gives a correct classification rate of 99.329%.

7.4.5 Tableware

Table 7.23 shows the results for classifying the Glass data set with patterns of the tableware class as the self set. The ALCs were trained with the negative selection method. The best classification result was obtained when IS=53 and W1=50 since the average number of ALCs was the lowest for all IS and W1 with an fNeg value of 0.000. An average number of 30.220 ALCs formed part of the active set of ALCs. An average number of 23.280 of the ALCs in the active set had

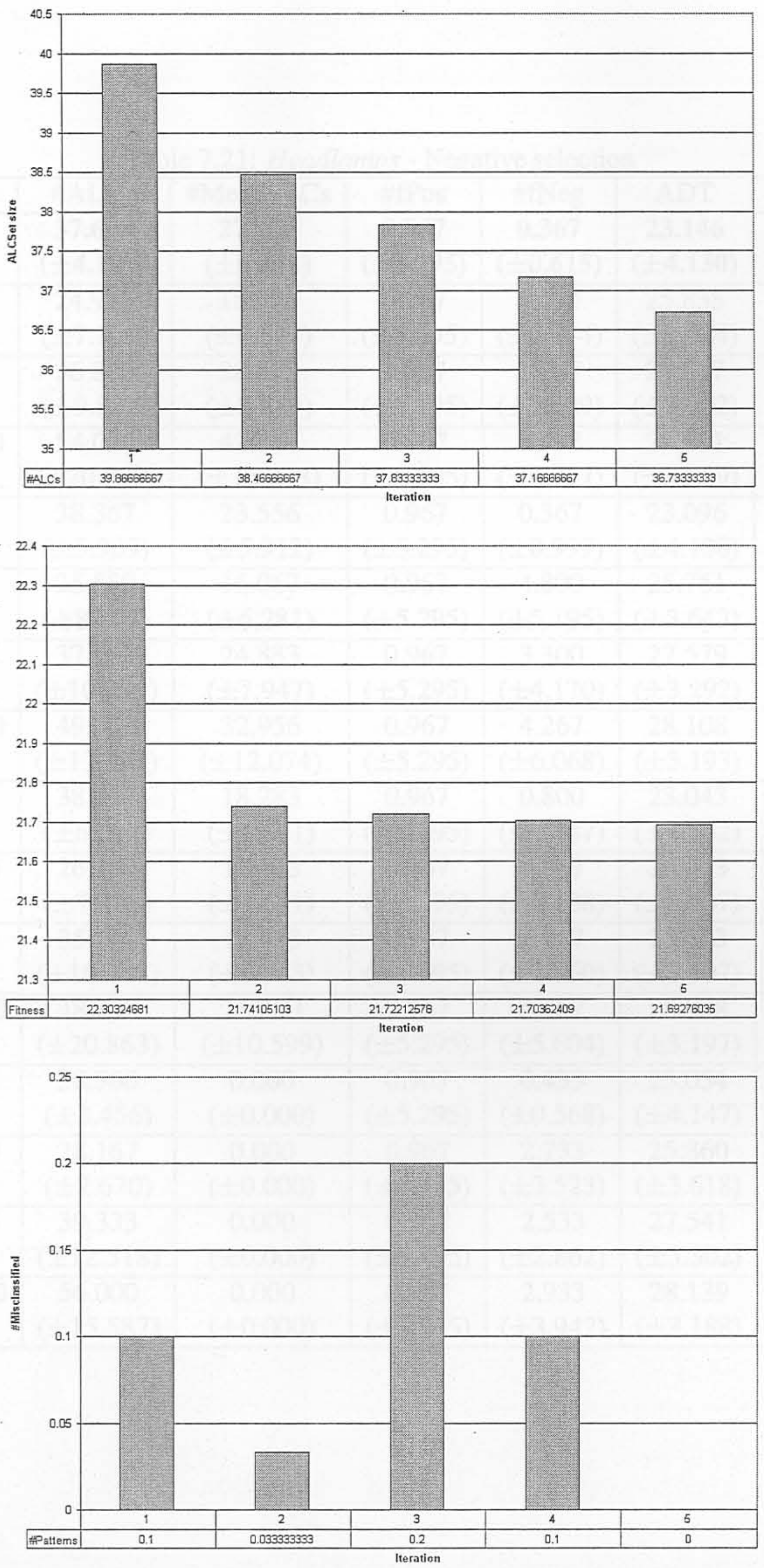


Figure 7.10: *Containers* - Positive selection with IS=53 and W1=25

Table 7.21: *Headlamps* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	**25**	**37.693 (±4.189)**	**27.133 (±6.224)**	**0.967 (±5.295)**	**0.367 (±0.615)**	**23.146 (±4.130)**	**21.739 (±0.160)**
	50	24.973 (±7.100)	18.300 (±6.504)	0.967 (±5.295)	5.633 (±7.374)	25.835 (±3.624)	19.205 (±0.663)
	75	36.840 (±9.887)	28.833 (±8.849)	0.967 (±5.295)	3.767 (±3.329)	27.537 (±3.302)	16.659 (±1.093)
	100	54.027 (±20.529)	42.747 (±16.873)	0.967 (±5.295)	4.533 (±6.917)	28.031 (±3.209)	15.318 (±0.736)
106	25	38.367 (±5.909)	23.556 (±5.912)	0.967 (±5.295)	0.367 (±0.999)	23.096 (±4.138)	21.762 (±0.151)
	50	25.589 (±8.129)	16.067 (±6.281)	0.967 (±5.295)	4.800 (±5.195)	25.761 (±3.643)	19.361 (±0.628)
	75	37.667 (±10.819)	24.833 (±7.947)	0.967 (±5.295)	3.300 (±4.170)	27.579 (±3.292)	16.575 (±1.099)
	100	49.633 (±17.670)	32.956 (±12.074)	0.967 (±5.295)	4.267 (±6.068)	28.108 (±3.193)	15.209 (±0.889)
160	25	38.667 (±6.692)	18.283 (±4.861)	0.967 (±5.295)	0.800 (±2.747)	23.043 (±4.152)	21.791 (±0.143)
	50	26.100 (±7.928)	12.483 (±4.551)	0.967 (±5.295)	3.867 (±5.138)	25.773 (±3.637)	19.333 (±0.635)
	75	35.267 (±10.948)	17.433 (±5.943)	0.967 (±5.295)	2.467 (±2.360)	27.513 (±3.307)	16.691 (±1.102)
	100	48.800 (±20.863)	24.300 (±10.599)	0.967 (±5.295)	5.667 (±5.604)	28.088 (±3.197)	15.107 (±0.642)
214	25	38.700 (±3.456)	0.000 (±0.000)	0.967 (±5.295)	0.433 (±0.568)	23.054 (±4.147)	21.812 (±0.136)
	50	28.167 (±7.670)	0.000 (±0.000)	0.967 (±5.295)	2.733 (±3.523)	25.860 (±3.618)	19.309 (±0.615)
	75	39.333 (±12.518)	0.000 (±0.000)	0.967 (±5.295)	2.533 (±2.862)	27.541 (±3.302)	16.659 (±1.126)
	100	56.000 (±15.587)	0.000 (±0.000)	0.967 (±5.295)	2.933 (±3.947)	28.139 (±3.188)	15.107 (±0.750)

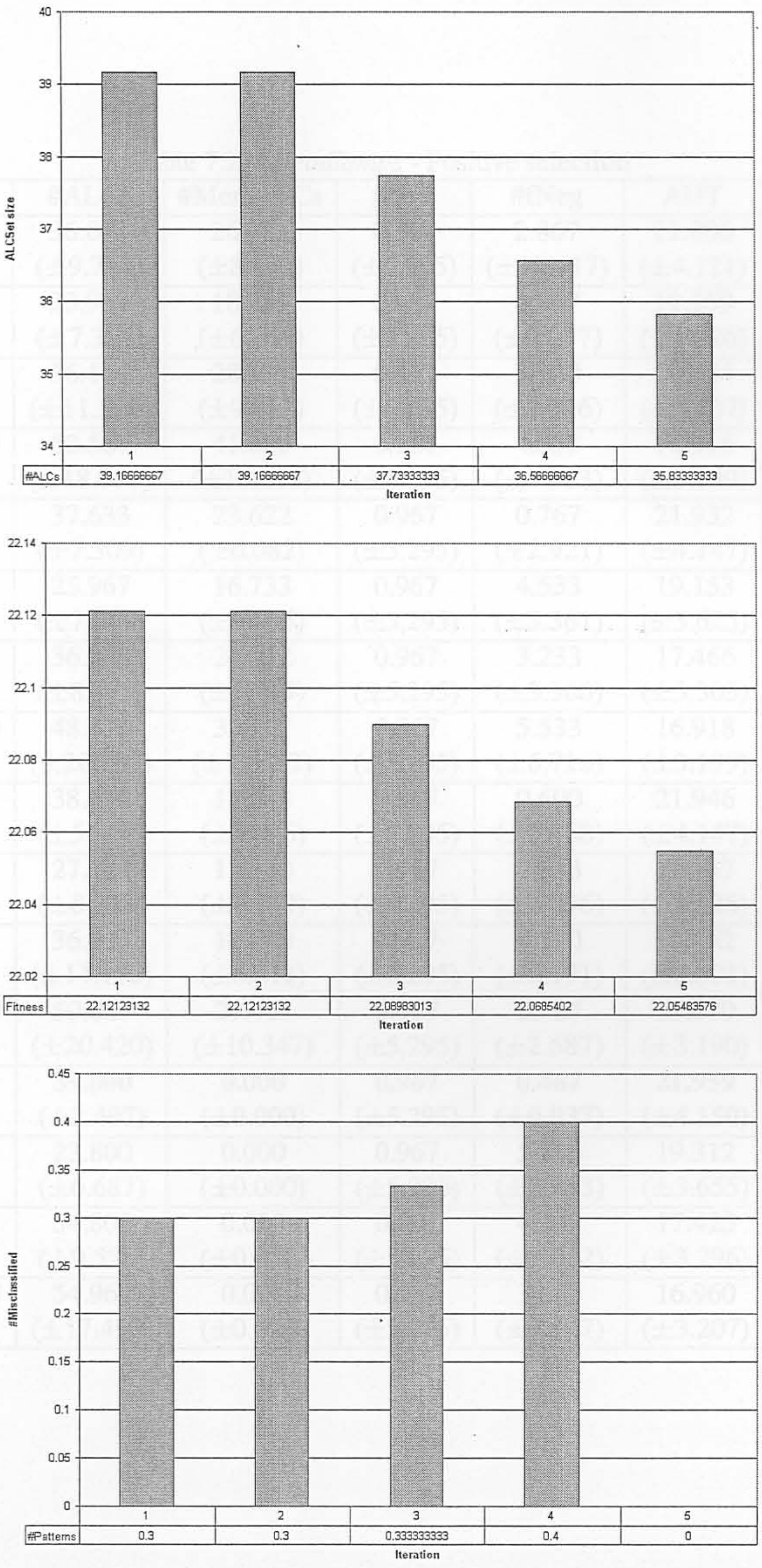


Figure 7.11: *Headlamps* - Negative selection with IS=53 and W1=25

Table 7.22: *Headlamps* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	36.033 (±9.754)	26.713 (±8.493)	0.967 (±5.295)	2.867 (±12.417)	21.806 (±4.121)	21.069 (±3.138)
	50	23.960 (±7.319)	18.253 (±6.349)	0.967 (±5.295)	4.267 (±5.037)	19.159 (±3.626)	18.637 (±2.683)
	75	36.140 (±11.528)	28.593 (±9.517)	0.967 (±5.295)	3.700 (±2.996)	17.484 (±3.307)	16.066 (±2.212)
	100	52.567 (±18.917)	41.680 (±15.268)	0.967 (±5.295)	4.033 (±4.803)	16.916 (±3.199)	14.688 (±2.115)
106	25	37.633 (±7.309)	23.622 (±6.082)	0.967 (±5.295)	0.767 (±2.921)	21.932 (±4.147)	21.278 (±2.604)
	50	25.967 (±7.917)	16.733 (±5.955)	0.967 (±5.295)	4.533 (±5.361)	19.153 (±3.625)	18.756 (±2.142)
	75	36.333 (±8.373)	24.022 (±5.919)	0.967 (±5.295)	3.233 (±3.360)	17.466 (±3.303)	16.198 (±1.685)
	100	48.378 (±20.339)	32.167 (±13.682)	0.967 (±5.295)	5.533 (±6.715)	16.918 (±3.199)	14.748 (±1.737)
160	25	38.450 (±5.207)	18.583 (±4.096)	0.967 (±5.295)	0.600 (±0.968)	21.946 (±4.147)	21.421 (±1.923)
	50	27.183 (±8.278)	13.300 (±4.739)	0.967 (±5.295)	2.633 (±3.296)	19.167 (±3.625)	18.966 (±1.475)
	75	36.433 (±11.116)	18.000 (±6.011)	0.967 (±5.295)	4.500 (±6.191)	17.452 (±3.301)	16.244 (±1.026)
	100	59.067 (±20.420)	29.467 (±10.347)	0.967 (±5.295)	2.767 (±2.687)	16.870 (±3.190)	14.834 (±1.061)
214	**25**	**39.000 (±2.407)**	**0.000 (±0.000)**	**0.967 (±5.295)**	**0.467 (±0.937)**	**21.959 (±4.150)**	**21.803 (±0.144)**
	50	23.800 (±6.687)	0.000 (±0.000)	0.967 (±5.295)	5.133 (±5.463)	19.312 (±3.655)	19.401 (±0.631)
	75	34.800 (±9.528)	0.000 (±0.000)	0.967 (±5.295)	4.233 (±3.812)	17.425 (±3.296)	16.565 (±1.143)
	100	54.967 (±17.490)	0.000 (±0.000)	0.967 (±5.295)	3.600 (±3.597)	16.960 (±3.207)	15.261 (±0.643)

Table 7.23: *Tableware* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	41.153 (±6.913)	31.407 (±8.113)	0.300 (±1.643)	0.000 (±0.000)	25.355 (±3.714)	21.451 (±0.218)
	50	**30.220 (±7.324)**	**23.280 (±7.316)**	**0.300 (±1.643)**	**0.000 (±0.000)**	**28.786 (±3.072)**	**18.268 (±0.824)**
	75	43.333 (±10.996)	34.373 (±9.781)	0.300 (±1.643)	0.000 (±0.000)	30.718 (±2.706)	15.494 (±1.326)
	100	68.833 (±21.849)	54.933 (±17.893)	0.300 (±1.643)	0.000 (±0.000)	31.425 (±2.568)	14.005 (±1.135)
106	25	41.533 (±3.652)	26.556 (±5.616)	0.300 (±1.643)	0.000 (±0.000)	25.241 (±3.735)	21.519 (±0.196)
	50	26.656 (±8.050)	17.022 (±6.242)	0.300 (±1.643)	0.033 (±0.183)	28.732 (±3.079)	18.285 (±0.823)
	75	44.400 (±14.628)	29.356 (±10.400)	0.300 (±1.643)	0.033 (±0.183)	30.738 (±2.698)	15.449 (±1.326)
	100	61.800 (±22.884)	41.089 (±15.551)	0.300 (±1.643)	0.067 (±0.254)	31.411 (±2.572)	13.934 (±0.711)
160	25	41.367 (±3.102)	19.833 (±4.075)	0.300 (±1.643)	0.000 (±0.000)	25.229 (±3.737)	21.521 (±0.193)
	50	27.433 (±6.912)	13.200 (±4.248)	0.300 (±1.643)	0.133 (±0.346)	28.732 (±3.078)	18.298 (±0.811)
	75	43.900 (±10.142)	21.767 (±5.741)	0.300 (±1.643)	0.000 (±0.000)	30.775 (±2.691)	15.417 (±1.322)
	100	66.033 (±19.817)	32.933 (±10.181)	0.300 (±1.643)	0.000 (±0.000)	31.442 (±2.564)	13.939 (±0.786)
214	25	42.267 (±2.791)	0.000 (±0.000)	0.300 (±1.643)	0.000 (±0.000)	25.211 (±3.741)	21.531 (±0.189)
	50	27.000 (±9.432)	0.000 (±0.000)	0.300 (±1.643)	0.067 (±0.254)	28.716 (±3.085)	18.313 (±0.815)
	75	39.400 (±10.559)	0.000 (±0.000)	0.300 (±1.643)	0.000 (±0.000)	30.799 (±2.688)	15.354 (±1.333)
	100	73.733 (±24.271)	0.000 (±0.000)	0.300 (±1.643)	0.000 (±0.000)	31.387 (±2.576)	14.064 (±0.924)

memory status. The ALCs misclassified an average of 0.000 patterns as tableware and 0.300 as not. Thus, $\#Misclassified = 0.000 + 0.300 = 0.300$ which gives a correct classification rate of 99.85%. Figure 7.12 shows that the average number of ALCs decreased from 30.233 to 30.2 and the average fitness of the ALC set decreased from 23.528 to 23.525 over five iterations. The average number of misclassifications increased from iteration one to iteration three and then decreased at iteration five.

Table 7.24 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=106 and W1=50 since the average number of ALCs was the lowest for all IS and W1 with an fNeg value of 0.000. An average number of 26.300 ALCs formed part of the active set of ALCs. An average number of 17.233 of the ALCs in the active set had memory status. The ALCs misclassified an average of 0.000 patterns as tableware and 0.300 as not. Thus, $\#Misclassified = 0.000 + 0.300 = 0.300$ which gives a correct classification rate of 99.85%. Figure 7.13 shows a decrease in the average number of ALCs and a decrease in the average fitness of the ALC set. The average number of misclassifications increased from iteration one to two and then drops to 0.0 in iteration three.

7.4.6 Vehicle-window-float

Table 7.25 shows the results for classifying the Glass data set with patterns of the vehicle-window-float class as the self set. The ALCs were trained with the negative selection method. The best classification result was obtained when IS=106 and W1=25. An average number of 39.989 ALCs formed part of the active set of ALCs. An average number of 25.289 of the ALCs in the active set had memory status. The ALCs misclassified an average of 2.267 patterns as vehicle-window-float and 0.567 as not. Thus, $\#Misclassified = 2.267 + 0.567 = 2.834$ which gives a correct classification rate of 98.675%. Figure 7.14 shows a decrease in the average number of ALCs, a decrease in the average fitness of the ALC set and a decrease in the average number of misclassified patterns over all the iterations.

Table 7.26 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=106 and W1=25. An average number of 39.967 ALCs formed part of the active set of ALCs. An average number of 25.933 of the ALCs in the active set had memory status. The ALCs misclassified an average of 2.000 patterns as vehicle-window-float and 0.567 as not. Thus, $\#Misclassified = 2.000 + 0.567 = 2.567$ which gives a correct classification rate of 98.800%. Figure 7.15 also shows a decrease in the average

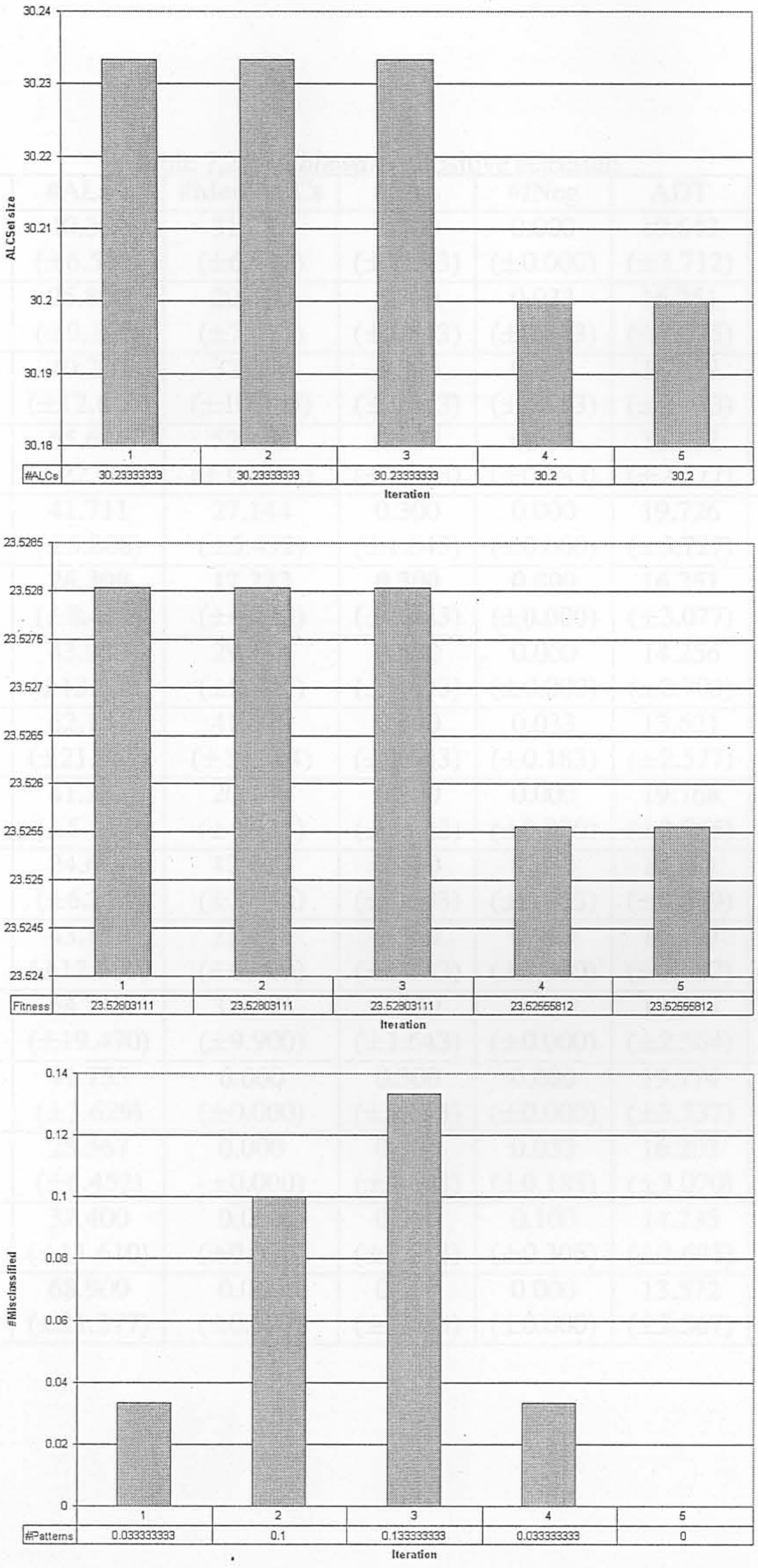


Figure 7.12: *Tableware* - Negative selection with IS=53 and W1=50

Table 7.24: *Tableware* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	40.380 (±6.503)	31.713 (±6.432)	0.300 (±1.643)	0.000 (±0.000)	19.643 (±3.712)	20.858 (±3.091)
	50	25.800 (±9.114)	20.420 (±7.743)	0.300 (±1.643)	0.033 (±0.183)	16.251 (±3.075)	17.656 (±2.496)
	75	40.207 (±12.660)	32.080 (±10.393)	0.300 (±1.643)	0.033 (±0.183)	14.227 (±2.693)	14.791 (±1.965)
	100	65.627 (±22.403)	52.453 (±18.065)	0.300 (±1.643)	0.033 (±0.183)	13.607 (±2.577)	13.558 (±1.898)
106	25	41.711 (±5.808)	27.144 (±5.492)	0.300 (±1.643)	0.000 (±0.000)	19.726 (±3.727)	21.005 (±2.552)
	50	**26.300 (±8.422)**	**17.233 (±6.229)**	**0.300 (±1.643)**	**0.000 (±0.000)**	**16.251 (±3.077)**	**17.786 (±1.966)**
	75	43.933 (±13.620)	29.156 (±9.485)	0.300 (±1.643)	0.000 (±0.000)	14.256 (±2.703)	14.945 (±1.450)
	100	62.144 (±21.347)	41.378 (±14.384)	0.300 (±1.643)	0.033 (±0.183)	13.621 (±2.577)	13.603 (±1.421)
160	25	41.233 (±5.211)	20.150 (±4.221)	0.300 (±1.643)	0.000 (±0.000)	19.768 (±3.735)	21.150 (±1.872)
	50	24.617 (±6.272)	12.033 (±3.796)	0.300 (±1.643)	0.033 (±0.183)	16.301 (±3.089)	17.970 (±1.300)
	75	43.150 (±12.229)	21.467 (±6.469)	0.300 (±1.643)	0.000 (±0.000)	14.197 (±2.687)	14.990 (±0.773)
	100	64.267 (±19.470)	32.083 (±9.900)	0.300 (±1.643)	0.000 (±0.000)	13.555 (±2.564)	13.647 (±0.870)
214	25	41.733 (±3.629)	0.000 (±0.000)	0.300 (±1.643)	0.000 (±0.000)	19.774 (±3.737)	21.535 (±0.187)
	50	25.567 (±6.452)	0.000 (±0.000)	0.300 (±1.643)	0.033 (±0.183)	16.203 (±3.070)	18.229 (±0.842)
	75	37.400 (±11.610)	0.000 (±0.000)	0.300 (±1.643)	0.100 (±0.305)	14.235 (±2.695)	15.331 (±1.349)
	100	68.900 (±21.377)	0.000 (±0.000)	0.300 (±1.643)	0.000 (±0.000)	13.572 (±2.567)	14.002 (±0.775)

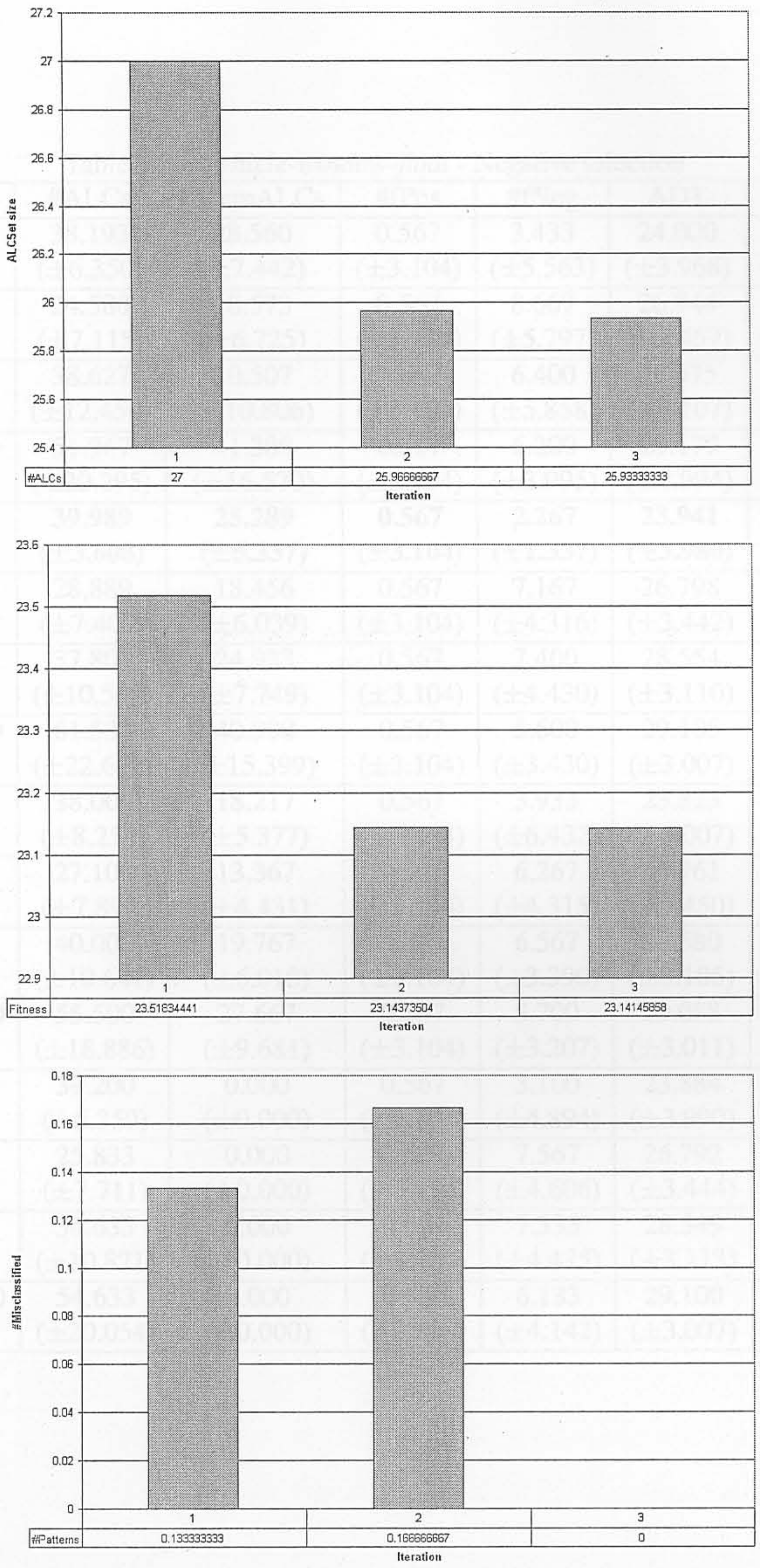
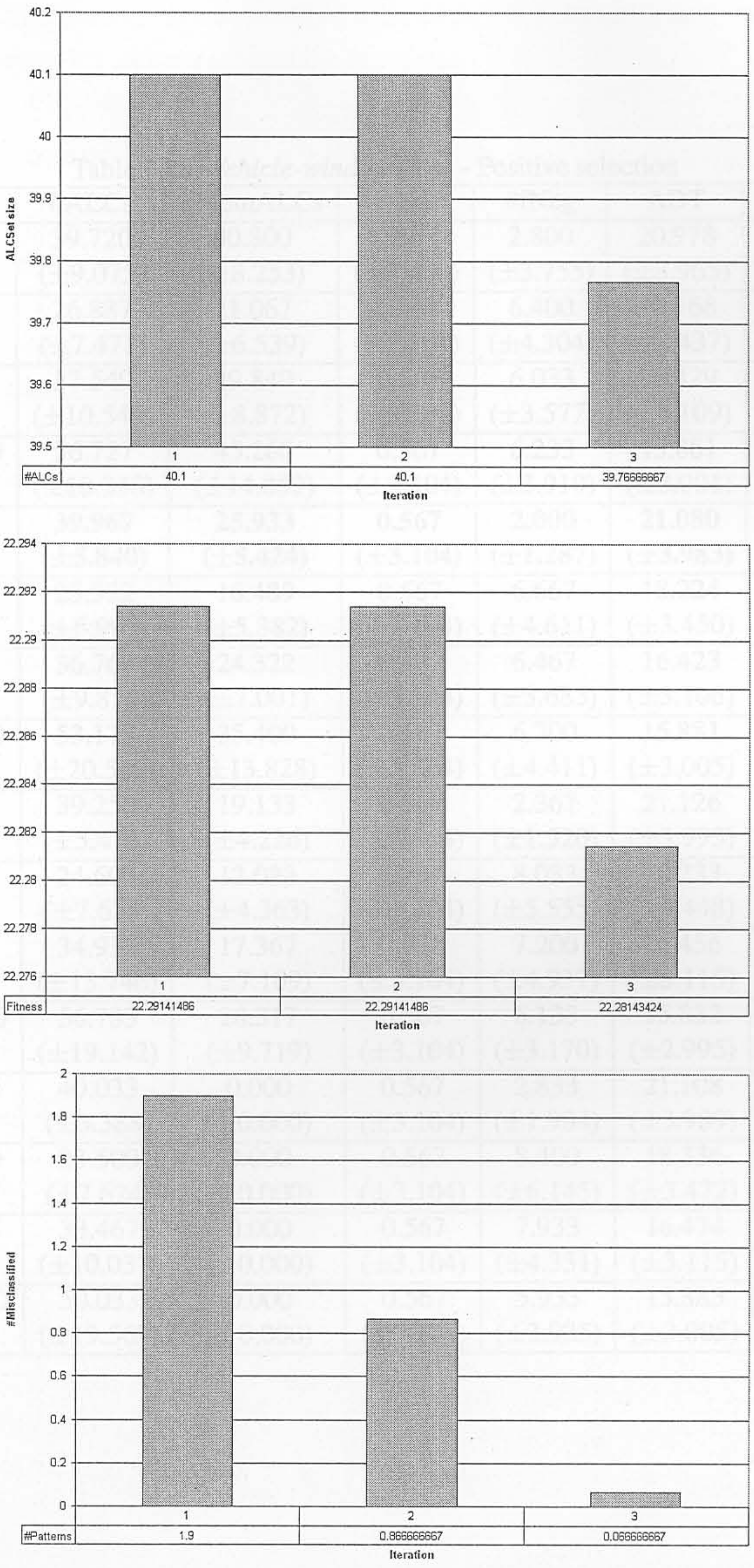


Figure 7.13: *Tableware* - Positive selection with IS=106 and W1=50

Table 7.25: *Vehicle-window-float* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	38.193 (±6.350)	28.560 (±7.442)	0.567 (±3.104)	3.433 (±5.563)	24.000 (±3.968)	21.687 (±0.226)
	50	24.580 (±7.115)	18.573 (±6.725)	0.567 (±3.104)	8.667 (±5.797)	26.744 (±3.457)	19.113 (±0.693)
	75	38.627 (±12.450)	30.507 (±10.806)	0.567 (±3.104)	6.400 (±5.858)	28.575 (±3.107)	16.350 (±1.163)
	100	51.967 (±20.295)	41.380 (±16.570)	0.567 (±3.104)	6.200 (±3.995)	29.179 (±2.994)	15.003 (±0.840)
106	**25**	**39.989 (±3.608)**	**25.289 (±5.337)**	**0.567 (±3.104)**	**2.267 (±1.337)**	**23.941 (±3.980)**	**21.737 (±0.153)**
	50	28.889 (±7.405)	18.456 (±6.039)	0.567 (±3.104)	7.167 (±4.316)	26.798 (±3.442)	19.191 (±0.646)
	75	37.800 (±10.535)	24.933 (±7.749)	0.567 (±3.104)	7.400 (±4.430)	28.554 (±3.110)	16.460 (±1.138)
	100	61.533 (±22.642)	40.900 (±15.399)	0.567 (±3.104)	5.600 (±3.430)	29.105 (±3.007)	15.114 (±0.733)
160	25	38.000 (±8.259)	18.217 (±5.377)	0.567 (±3.104)	3.933 (±6.432)	23.823 (±4.007)	21.754 (±0.147)
	50	27.100 (±7.893)	13.367 (±4.431)	0.567 (±3.104)	6.267 (±4.315)	26.761 (±3.450)	19.178 (±0.622)
	75	40.000 (±10.641)	19.767 (±6.018)	0.567 (±3.104)	6.567 (±3.390)	28.580 (±3.105)	16.440 (±1.149)
	100	55.500 (±18.886)	27.667 (±9.681)	0.567 (±3.104)	5.700 (±3.207)	29.088 (±3.011)	15.073 (±0.531)
214	25	39.200 (±6.359)	0.000 (±0.000)	0.567 (±3.104)	3.100 (±4.894)	23.884 (±3.990)	21.748 (±0.153)
	50	25.833 (±7.711)	0.000 (±0.000)	0.567 (±3.104)	7.567 (±4.606)	26.792 (±3.444)	19.150 (±0.670)
	75	34.633 (±10.871)	0.000 (±0.000)	0.567 (±3.104)	7.333 (±4.475)	28.549 (±3.113)	16.472 (±1.136)
	100	54.633 (±20.054)	0.000 (±0.000)	0.567 (±3.104)	6.133 (±4.142)	29.100 (±3.007)	15.081 (±0.626)



Iteration	1	2	3
#ALCs	40.1	40.1	39.76666667

Iteration	1	2	3
Fitness	22.29141486	22.29141486	22.28143424

Iteration	1	2	3
#Patterns	1.9	0.866666667	0.066666667

Figure 7.14: *Vehicle-window-float* - Negative selection with IS=106 and W1=25

Table 7.26: *Vehicle-window-float* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
53	25	39.720 (±9.075)	30.800 (±8.253)	0.567 (±3.104)	2.800 (±3.755)	20.978 (±3.965)	21.100 (±3.138)
	50	26.887 (±7.471)	21.067 (±6.539)	0.567 (±3.104)	6.400 (±4.304)	18.168 (±3.437)	18.547 (±2.661)
	75	37.540 (±10.544)	29.840 (±8.872)	0.567 (±3.104)	6.033 (±3.577)	16.429 (±3.109)	15.877 (±2.172)
	100	56.727 (±18.358)	45.260 (±14.859)	0.567 (±3.104)	6.233 (±3.910)	15.861 (±3.001)	14.547 (±2.072)
106	**25**	**39.967 (±5.840)**	**25.933 (±5.424)**	**0.567 (±3.104)**	**2.000 (±1.287)**	**21.080 (±3.983)**	**21.239 (±2.596)**
	50	25.322 (±6.997)	16.489 (±5.382)	0.567 (±3.104)	6.667 (±4.611)	18.224 (±3.450)	18.622 (±2.117)
	75	36.767 (±9.812)	24.322 (±7.001)	0.567 (±3.104)	6.467 (±3.683)	16.423 (±3.106)	15.971 (±1.629)
	100	53.178 (±20.539)	35.400 (±13.828)	0.567 (±3.104)	6.700 (±4.411)	15.881 (±3.005)	14.646 (±1.675)
160	25	39.250 (±5.415)	19.133 (±4.226)	0.567 (±3.104)	2.367 (±1.520)	21.126 (±3.993)	21.370 (±1.913)
	50	24.600 (±7.623)	12.033 (±4.363)	0.567 (±3.104)	8.033 (±5.555)	18.233 (±3.448)	18.762 (±1.439)
	75	34.933 (±13.746)	17.367 (±7.109)	0.567 (±3.104)	7.200 (±4.937)	16.456 (±3.115)	16.024 (±1.007)
	100	56.733 (±19.142)	28.317 (±9.719)	0.567 (±3.104)	6.133 (±3.170)	15.832 (±2.995)	14.666 (±1.094)
214	25	40.033 (±3.388)	0.000 (±0.000)	0.567 (±3.104)	2.833 (±1.984)	21.108 (±3.989)	21.758 (±0.146)
	50	23.500 (±7.624)	0.000 (±0.000)	0.567 (±3.104)	8.400 (±6.145)	18.336 (±3.472)	19.264 (±0.658)
	75	33.467 (±10.037)	0.000 (±0.000)	0.567 (±3.104)	7.933 (±4.331)	16.474 (±3.115)	16.464 (±1.164)
	100	50.033 (±19.567)	0.000 (±0.000)	0.567 (±3.104)	5.933 (±2.935)	15.885 (±3.005)	15.046 (±0.698)

number of ALCs, a decrease in the average fitness of the ALC set and a decrease in the average number of misclassified patterns over all the iterations.

7.4.7 Conclusion: Glass

The above results show that in most cases, except in the case of the *building-window-float* class as self, the positive selection method had better classification results than the negative selection method, though the parameter settings for IS and W1 were different for each case. In cases where the correct classification results were the same, as is the case with *containers* and *tableware* as self sets, the positive selection method had better performance compared to the negative selection method since the average number of ALCs in the active set was less for positive selection than for negative selection.

7.5 Car Evaluation

The car evaluation data set was derived from a simple hierarchical decision model that was developed by [8]. The model evaluates cars according to three concept structures, namely overall price, technical characteristics and comfort. The overall price concept is related to the buying price and price of maintenance, technical characteristics is related to the estimated safety of the car and the comfort which is related to the number of doors, the capacity in terms of persons to carry and the size of the luggage boot. The car evaluation data set contains examples with the structural concepts removed and directly relates a car to the six input attributes. All of these attributes are nominally valued. Since the car database has underlying concept structures, the database may be particularly useful for testing constructive induction and structure discovery methods. The car evaluation data set consists of 1728 patterns that are distributed between 4 car classes. These classes are acceptable, good, unacceptable and very good. 1210 patterns are of the unacceptable class, 384 are of the acceptable class, 69 are of the good class and 65 are of the very good class. The patterns were converted to binary strings of length 12.

7.5.1 Acceptable

Table 7.27 shows that with IS=864 and W1=50 the lowest misclassification of 916.546 patterns ($\#Misclassified = 915.113 + 1.433 = 916.546$) was achieved when training the ALCs with negative selection on patterns of the acceptable class as self. This gives a correct classification of 46.959% with an average number of 11.967 ALCs in the active set. An average number of 5.900

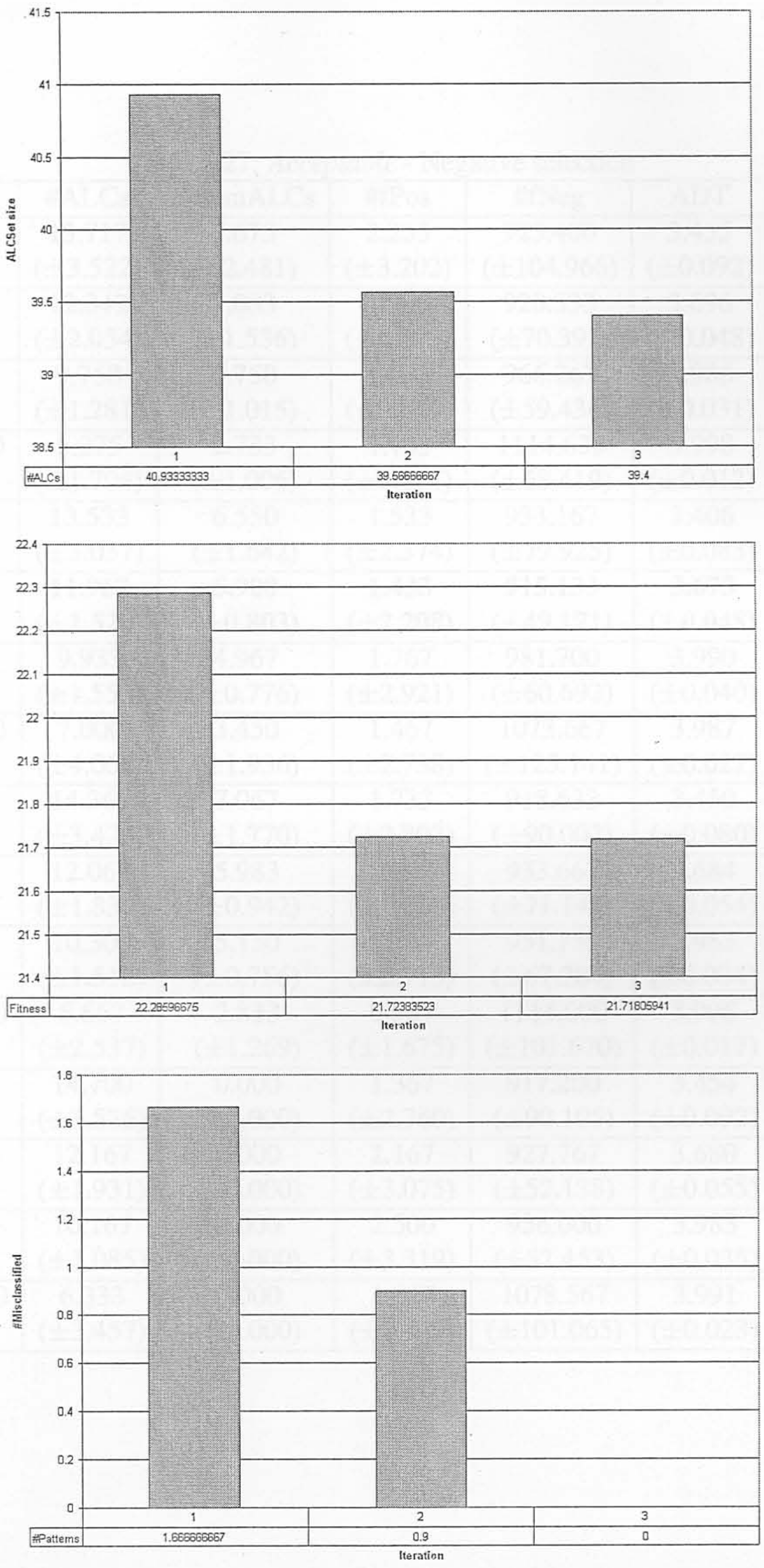


Figure 7.15: *Vehicle-window-float* - Positive selection with IS=106 and W1=25

Table 7.27: *Acceptable* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	13.717 (±3.522)	7.675 (±2.481)	2.233 (±3.202)	923.400 (±104.966)	3.455 (±0.092)	5.900 (±0.061)
	50	12.342 (±2.054)	7.083 (±1.536)	1.700 (±2.322)	920.533 (±70.391)	3.696 (±0.048)	5.732 (±0.063)
	75	9.758 (±1.281)	5.750 (±1.015)	1.533 (±2.849)	966.267 (±59.436)	3.986 (±0.031)	4.950 (±0.107)
	100	5.275 (±1.794)	2.733 (±1.006)	1.733 (±2.864)	1114.633 (±68.419)	3.998 (±0.012)	3.979 (±0.499)
864	25	13.533 (±3.037)	6.550 (±1.642)	1.533 (±2.374)	933.167 (±79.925)	3.406 (±0.083)	5.944 (±0.020)
	50	**11.967 (±1.520)**	**5.900 (±0.803)**	**1.433 (±2.208)**	**915.133 (±49.171)**	**3.673 (±0.048)**	**5.747 (±0.037)**
	75	9.933 (±1.552)	4.967 (±0.776)	1.767 (±2.921)	981.700 (±60.692)	3.990 (±0.040)	4.953 (±0.100)
	100	7.000 (±4.068)	3.450 (±1.936)	1.467 (±2.738)	1073.667 (±125.141)	3.987 (±0.027)	4.184 (±0.434)
1296	25	14.367 (±3.429)	7.067 (±1.770)	1.733 (±2.803)	918.633 (±90.002)	3.450 (±0.080)	5.949 (±0.018)
	50	12.067 (±1.837)	5.983 (±0.942)	1.667 (±2.708)	933.667 (±71.145)	3.684 (±0.054)	5.741 (±0.058)
	75	10.300 (±1.512)	5.150 (±0.756)	1.500 (±2.713)	951.733 (±67.284)	3.983 (±0.034)	4.967 (±0.098)
	100	5.667 (±2.537)	2.833 (±1.269)	0.567 (±1.675)	1114.000 (±101.670)	3.996 (±0.017)	3.974 (±0.582)
1728	25	14.700 (±3.525)	0.000 (±0.000)	1.367 (±2.760)	917.200 (±90.105)	3.454 (±0.092)	5.944 (±0.019)
	50	12.167 (±1.931)	0.000 (±0.000)	2.167 (±3.075)	927.767 (±52.138)	3.680 (±0.055)	5.747 (±0.053)
	75	10.167 (±1.085)	0.000 (±0.000)	2.500 (±3.319)	956.000 (±57.453)	3.983 (±0.035)	4.975 (±0.100)
	100	6.333 (±3.457)	0.000 (±0.000)	1.067 (±2.449)	1078.567 (±101.065)	3.991 (±0.023)	4.157 (±0.520)

of the ALCs in the active set had memory status.

Table 7.28 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=864 and W1=25. An average number of 15.533 ALCs formed part of the active set of ALCs. An average number of 7.567 of the ALCs in the active set had memory status. The ALCs misclassified an average of 897.733 patterns as acceptable and 1.367 as not. Thus, $\#Misclassified = 897.733 + 1.367 = 899.100$ which gives a correct classification rate of 47.968%.

7.5.2 Good

The classification results for training the ALCs with negative selection on patterns of the good class as self is shown in table 7.29. The lowest misclassification of 230.233 patterns falsely as good and 0.433 as not ($\#Misclassified = 230.233 + 0.433 = 230.666$) was achieved with IS=432 and W1=50. This gives a correct classification of 86.651% with an average number of 17.642 ALCs in the active set of ALCs. The average number of ALCs with memory status in the active set was 12.675. Figure 7.16 shows that the average number of ALCs decreased over the iterations from 17.766 to 17.333 and the average fitness of the ALC set also decreased from 5.085 to 5.058 over the iterations. The number of misclassified patterns had an average decrease from 68.433 to 48.9.

Table 7.30 shows the results when the ALCs were trained with the positive selection method. The best classification result was achieved when IS=1296 and W1=50 which gave an average number of 16.833 ALCs in the active set of ALCs. An average number of 8.383 of the ALCs in the active set had memory status. The ALCs classified 243.000 patterns falsely as good and 0.367 as not, which gave a misclassification of 243.367 patterns ($\#Misclassified = 243.000 + 0.367 = 243.367$) and a correct classification rate of 85.916%.

7.5.3 Unacceptable

Table 7.31 shows that with IS=864 and W1=75 the lowest misclassification of 433.800 patterns ($\#Misclassified = 426.733 + 7.067 = 433.800$) was achieved when training the ALCs with negative selection on patterns of the unacceptable class as self. The ALCs misclassified an average of 426.733 patterns as unacceptable and 7.067 as not. This gives a correct classification of 74.895% with an average number of 11.567 ALCs in the active set. An average number of 5.783 of the

Table 7.28: *Acceptable* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	13.508 (±3.531)	7.550 (±2.559)	1.900 (±2.510)	927.267 (±91.816)	8.537 (±0.090)	5.896 (±0.063)
	50	11.908 (±1.725)	6.717 (±1.243)	1.867 (±2.825)	928.567 (±71.491)	8.315 (±0.039)	5.727 (±0.063)
	75	9.508 (±1.172)	5.375 (±0.801)	1.867 (±2.933)	965.667 (±59.833)	8.008 (±0.023)	4.923 (±0.069)
	100	5.500 (±2.288)	3.008 (±1.505)	1.100 (±2.496)	1075.233 (±84.297)	8.002 (±0.008)	3.934 (±0.446)
864	**25**	**15.533 (±3.767)**	**7.567 (±1.870)**	**1.367 (±2.157)**	**897.733 (±88.295)**	**8.542 (±0.070)**	**5.950 (±0.012)**
	50	11.667 (±1.373)	5.783 (±0.703)	1.500 (±2.177)	933.700 (±54.525)	8.318 (±0.053)	5.736 (±0.048)
	75	10.100 (±1.213)	5.050 (±0.607)	1.867 (±2.886)	944.933 (±57.878)	8.027 (±0.046)	5.001 (±0.139)
	100	5.667 (±2.537)	2.833 (±1.269)	1.367 (±3.068)	1108.967 (±95.542)	8.004 (±0.017)	3.921 (±0.539)
1296	25	14.333 (±3.745)	7.050 (±1.882)	2.233 (±2.712)	922.100 (±92.388)	8.555 (±0.089)	5.944 (±0.027)
	50	11.900 (±1.494)	5.933 (±0.763)	2.367 (±3.347)	941.367 (±60.738)	8.316 (±0.053)	5.750 (±0.046)
	75	10.167 (±1.289)	5.083 (±0.644)	2.333 (±3.336)	941.767 (±54.185)	8.008 (±0.026)	4.965 (±0.092)
	100	5.000 (±0.000)	2.500 (±0.000)	0.900 (±2.074)	1135.233 (±68.284)	8.000 (±0.000)	3.840 (±0.554)
1728	25	14.067 (±3.463)	0.000 (±0.000)	1.633 (±2.710)	936.267 (±83.779)	8.562 (±0.070)	5.943 (±0.024)
	50	12.467 (±2.345)	0.000 (±0.000)	1.633 (±2.619)	917.800 (±59.883)	8.306 (±0.043)	5.750 (±0.053)
	75	10.300 (±1.088)	0.000 (±0.000)	1.233 (±2.528)	954.800 (±48.170)	8.020 (±0.041)	4.987 (±0.090)
	100	6.667 (±3.790)	0.000 (±0.000)	1.000 (±2.729)	1112.967 (±119.664)	8.011 (±0.025)	3.838 (±0.672)

Table 7.29: *Good* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	17.275 (±3.736)	10.683 (±2.391)	0.467 (±1.502)	311.233 (±94.128)	4.562 (±0.105)	5.725 (±0.090)
	50	**17.642 (±4.835)**	**12.675 (±3.657)**	**0.433 (±1.832)**	**230.233 (±75.159)**	**5.320 (±0.086)**	**4.836 (±0.135)**
	75	11.158 (±1.765)	8.358 (±1.327)	0.333 (±1.470)	378.533 (±58.707)	5.910 (±0.082)	3.291 (±0.339)
	100	8.267 (±5.159)	6.200 (±3.869)	0.133 (±0.730)	495.200 (±107.094)	5.977 (±0.036)	2.614 (±0.304)
864	25	18.733 (±3.759)	9.117 (±1.888)	0.467 (±1.833)	296.600 (±88.853)	4.463 (±0.064)	5.853 (±0.030)
	50	17.500 (±4.265)	8.750 (±2.132)	0.400 (±1.476)	244.933 (±74.043)	5.305 (±0.085)	4.870 (±0.101)
	75	10.333 (±1.729)	5.167 (±0.864)	0.367 (±1.474)	395.033 (±62.126)	5.930 (±0.076)	3.232 (±0.338)
	100	7.667 (±4.498)	3.833 (±2.249)	0.133 (±0.730)	555.633 (±160.979)	5.982 (±0.030)	2.511 (±0.474)
1296	25	18.600 (±3.001)	9.067 (±1.654)	0.433 (±1.501)	292.033 (±79.156)	4.447 (±0.104)	5.850 (±0.025)
	50	16.733 (±4.593)	8.367 (±2.297)	0.433 (±1.832)	244.467 (±75.462)	5.296 (±0.086)	4.862 (±0.121)
	75	10.900 (±1.845)	5.450 (±0.922)	0.367 (±1.474)	395.600 (±61.482)	5.931 (±0.064)	3.220 (±0.305)
	100	9.000 (±4.983)	4.500 (±2.491)	0.267 (±1.461)	511.267 (±167.143)	5.973 (±0.033)	2.583 (±0.498)
1728	25	18.233 (±3.674)	0.000 (±0.000)	0.500 (±1.526)	300.800 (±110.290)	4.453 (±0.137)	5.848 (±0.028)
	50	17.600 (±4.546)	0.000 (±0.000)	0.533 (±1.852)	244.567 (±66.496)	5.319 (±0.065)	4.849 (±0.120)
	75	10.167 (±1.289)	0.000 (±0.000)	0.333 (±1.470)	409.600 (±58.785)	5.948 (±0.060)	3.159 (±0.287)
	100	8.000 (±4.661)	0.000 (±0.000)	0.167 (±0.747)	518.967 (±140.424)	5.980 (±0.031)	2.663 (±0.316)

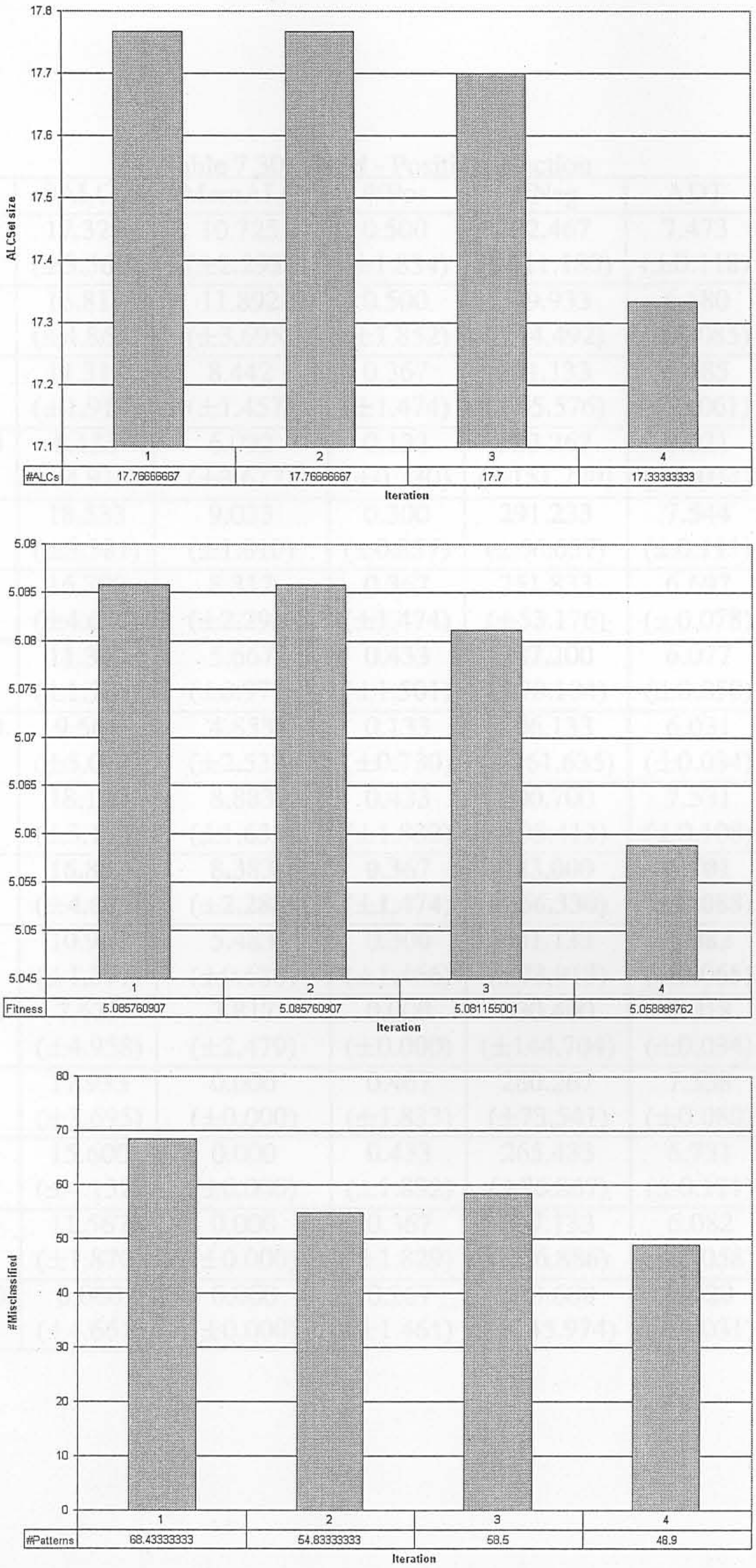


Figure 7.16: *Good* - Negative selection with IS=432 and W1=50

Table 7.30: *Good* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	17.325 (±3.366)	10.725 (±2.293)	0.500 (±1.834)	322.467 (±111.180)	7.473 (±0.118)	5.775 (±0.054)
	50	16.817 (±4.862)	11.892 (±3.695)	0.500 (±1.852)	249.933 (±74.492)	6.680 (±0.085)	4.833 (±0.103)
	75	11.317 (±1.917)	8.442 (±1.457)	0.367 (±1.474)	391.133 (±65.576)	6.085 (±0.061)	3.276 (±0.290)
	100	8.133 (±4.918)	6.092 (±3.677)	0.133 (±0.730)	533.267 (±151.720)	6.021 (±0.034)	2.608 (±0.415)
864	25	18.533 (±3.381)	9.033 (±1.810)	0.300 (±0.837)	291.233 (±66.657)	7.544 (±0.111)	5.846 (±0.028)
	50	16.700 (±4.617)	8.317 (±2.291)	0.367 (±1.474)	251.833 (±53.176)	6.697 (±0.078)	4.882 (±0.125)
	75	11.333 (±1.953)	5.667 (±0.977)	0.433 (±1.501)	387.200 (±70.104)	6.077 (±0.050)	3.257 (±0.307)
	100	9.667 (±5.074)	4.833 (±2.537)	0.133 (±0.730)	506.133 (±161.635)	6.031 (±0.034)	2.606 (±0.538)
1296	25	18.100 (±3.155)	8.883 (±1.633)	0.433 (±1.832)	300.700 (±93.412)	7.531 (±0.108)	5.849 (±0.029)
	50	**16.833 (±4.609)**	**8.383 (±2.288)**	**0.367 (±1.474)**	**243.000 (±66.330)**	**6.701 (±0.088)**	**4.874 (±0.085)**
	75	10.967 (±1.377)	5.483 (±0.688)	0.300 (±1.466)	401.133 (±73.913)	6.082 (±0.065)	3.238 (±0.283)
	100	7.633 (±4.958)	3.817 (±2.479)	0.000 (±0.000)	530.400 (±144.704)	6.018 (±0.034)	2.632 (±0.392)
1728	25	17.933 (±3.695)	0.000 (±0.000)	0.467 (±1.833)	280.267 (±73.541)	7.538 (±0.080)	5.851 (±0.028)
	50	15.600 (±4.132)	0.000 (±0.000)	0.433 (±1.832)	265.433 (±76.867)	6.731 (±0.111)	4.890 (±0.115)
	75	11.567 (±1.870)	0.000 (±0.000)	0.367 (±1.829)	387.133 (±56.886)	6.082 (±0.058)	3.250 (±0.321)
	100	8.000 (±4.661)	0.000 (±0.000)	0.267 (±1.461)	513.600 (±145.974)	6.020 (±0.031)	2.562 (±0.427)

Table 7.31: *Unacceptable* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	13.033 (±3.770)	5.533 (±1.682)	7.533 (±5.734)	450.300 (±29.880)	3.042 (±0.235)	5.119 (±0.440)
	50	15.300 (±3.723)	7.808 (±2.472)	11.667 (±5.990)	428.767 (±31.704)	3.468 (±0.195)	4.672 (±0.344)
	75	12.008 (±4.186)	6.808 (±3.145)	6.267 (±6.863)	429.033 (±32.505)	4.060 (±0.212)	2.942 (±0.347)
	100	9.983 (±9.282)	5.900 (±6.833)	3.600 (±5.512)	456.533 (±39.923)	4.100 (±0.250)	2.191 (±0.457)
864	25	17.617 (±3.854)	6.733 (±1.524)	10.033 (±5.346)	443.133 (±30.016)	2.773 (±0.156)	5.602 (±0.157)
	50	14.550 (±3.171)	6.517 (±1.600)	10.667 (±5.933)	432.600 (±34.523)	3.342 (±0.179)	4.926 (±0.297)
	75	**11.567 (±2.622)**	**5.783 (±1.311)**	**7.067 (±6.746)**	**426.733 (±33.595)**	**4.044 (±0.133)**	**3.005 (±0.241)**
	100	10.600 (±9.471)	5.283 (±4.728)	3.000 (±5.766)	452.867 (±44.584)	4.068 (±0.227)	2.276 (±0.423)
1296	25	14.883 (±4.413)	6.050 (±1.945)	8.833 (±6.114)	449.067 (±33.783)	2.794 (±0.177)	5.515 (±0.243)
	50	15.100 (±3.294)	7.083 (±1.727)	12.033 (±5.887)	435.967 (±32.905)	3.325 (±0.182)	4.956 (±0.249)
	75	11.767 (±2.582)	5.867 (±1.273)	7.133 (±6.872)	428.800 (±33.455)	4.042 (±0.161)	3.034 (±0.284)
	100	8.867 (±8.055)	4.433 (±4.027)	3.067 (±5.558)	455.600 (±46.120)	4.086 (±0.215)	2.275 (±0.410)
1728	25	18.100 (±4.413)	0.000 (±0.000)	10.767 (±6.484)	443.367 (±32.557)	2.536 (±0.088)	5.854 (±0.035)
	50	14.333 (±2.905)	0.000 (±0.000)	10.733 (±5.789)	438.500 (±34.400)	3.230 (±0.147)	5.129 (±0.181)
	75	12.000 (±3.151)	0.000 (±0.000)	7.167 (±6.869)	428.533 (±31.059)	4.038 (±0.184)	3.033 (±0.341)
	100	8.567 (±7.592)	0.000 (±0.000)	4.533 (±5.964)	456.800 (±43.608)	4.076 (±0.199)	2.327 (±0.397)

ALCs in the active set had memory status.

Table 7.32 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=1296 and W1=75. An average number of 12.400 ALCs formed part of the active set of ALCs. An average number of 6.200 of the ALCs in the active set had memory status. The ALCs misclassified an average of 425.233 patterns as unacceptable and 6.467 as not which gave a misclassification of 431.700 patterns ($\#Misclassified = 425.233 + 6.467 = 431.700$) and a correct classification rate of 75.017%.

7.5.4 Very Good

Table 7.33 shows the results for classifying the Car evaluation data set with patterns of the very good class as the self set. The ALCs were trained with the negative selection method. The best classification result was obtained when IS=432 and W1=50. An average number of 17.992 ALCs formed part of the active set of ALCs. An average number of 13.200 of the ALCs in the active set had memory status. The ALCs misclassified an average of 129.133 patterns as very good and 0.333 as not. Thus, $\#Misclassified = 129.133 + 0.333 = 129.466$ which gives a correct classification rate of 92.507%. Figure 7.17 shows a decrease in the average number of ALCs over the iterations from 18.1 to 17.8. The average fitness of the ALC set also decreased from 5.133 to 5.114. The average number of misclassified patterns decreased from iteration one to two, but then increased at iteration four.

Table 7.32 shows the results when the ALCs were trained with the positive selection method. The best classification result was obtained when IS=864 and W1=50. An average number of 18.233 ALCs formed part of the active set of ALCs. An average number of 9.067 of the ALCs in the active set had memory status. The ALCs misclassified an average of 130.033 patterns as unacceptable and 0.333 as not. Thus, $\#Misclassified = 130.033 + 0.333 = 130.366$ which gives a correct classification rate of 92.455%.

7.5.5 Conclusion: Car Evaluation

The above results show that different parameter settings for IS and W1 are necessary to obtain the best classification results for different classes as self. When comparing the best results of the above classes from both negative and positive selection as training methods, the average HD between the ALCs in the active set for the *acceptable* class as self is the highest for all the classes

Table 7.32: *Unacceptable* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	14.917 (±4.141)	6.500 (±2.121)	9.500 (±5.264)	447.200 (±30.398)	9.077 (±0.176)	5.267 (±0.310)
	50	13.258 (±3.529)	6.417 (±2.364)	10.600 (±6.360)	436.867 (±38.115)	8.577 (±0.194)	4.681 (±0.331)
	75	11.283 (±2.683)	6.158 (±2.148)	7.200 (±6.504)	429.867 (±33.360)	7.937 (±0.178)	2.982 (±0.323)
	100	10.292 (±10.352)	6.092 (±7.629)	3.700 (±6.221)	453.167 (±46.268)	7.920 (±0.216)	2.271 (±0.412)
864	25	15.800 (±4.656)	5.300 (±1.648)	7.733 (±5.401)	451.400 (±28.165)	9.513 (±0.111)	5.874 (±0.035)
	50	15.767 (±4.014)	6.767 (±1.851)	11.767 (±6.290)	433.967 (±35.004)	8.770 (±0.107)	5.147 (±0.121)
	75	12.067 (±3.999)	6.017 (±1.941)	5.400 (±7.040)	427.800 (±32.190)	7.933 (±0.203)	2.967 (±0.367)
	100	9.867 (±11.933)	4.933 (±5.966)	3.567 (±5.685)	459.800 (±47.640)	7.900 (±0.239)	2.236 (±0.399)
1296	25	19.633 (±3.961)	7.367 (±1.676)	10.067 (±6.203)	438.133 (±31.181)	9.461 (±0.099)	5.860 (±0.037)
	50	15.633 (±3.873)	7.100 (±1.927)	11.700 (±6.204)	430.333 (±38.301)	8.769 (±0.109)	5.148 (±0.122)
	75	**12.400 (±4.407)**	**6.200 (±2.203)**	**6.467 (±7.281)**	**425.233 (±36.141)**	**7.956 (±0.154)**	**3.034 (±0.245)**
	100	9.500 (±10.153)	4.750 (±5.077)	3.100 (±5.592)	457.100 (±42.542)	7.923 (±0.218)	2.255 (±0.409)
1728	25	16.833 (±4.836)	0.000 (±0.000)	8.933 (±5.426)	449.467 (±32.126)	9.492 (±0.103)	5.862 (±0.029)
	50	15.000 (±3.677)	0.000 (±0.000)	9.500 (±6.318)	431.400 (±35.091)	8.743 (±0.121)	5.102 (±0.153)
	75	12.167 (±4.395)	0.000 (±0.000)	6.333 (±6.825)	425.467 (±34.767)	7.945 (±0.178)	2.975 (±0.244)
	100	10.267 (±10.044)	0.000 (±0.000)	3.200 (±5.561)	449.433 (±48.340)	7.937 (±0.179)	2.357 (±0.340)

Table 7.33: *Very Good* - Negative selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	17.650 (±2.573)	10.983 (±1.838)	0.367 (±0.928)	157.833 (±49.543)	4.750 (±0.117)	5.681 (±0.088)
	50	**17.992 (±5.647)**	**13.200 (±4.251)**	**0.333 (±0.884)**	**129.133 (±45.808)**	**5.719 (±0.113)**	**4.532 (±0.191)**
	75	9.633 (±0.850)	7.225 (±0.638)	0.333 (±0.922)	193.900 (±36.771)	6.016 (±0.125)	3.972 (±0.174)
	100	6.667 (±4.611)	5.000 (±3.458)	0.233 (±0.626)	370.933 (±125.691)	6.021 (±0.156)	3.047 (±0.482)
864	25	18.683 (±3.573)	8.817 (±1.836)	0.400 (±0.932)	163.433 (±48.846)	4.627 (±0.108)	5.801 (±0.028)
	50	17.900 (±4.581)	8.950 (±2.291)	0.300 (±0.877)	129.900 (±49.694)	5.707 (±0.083)	4.562 (±0.148)
	75	9.700 (±0.988)	4.850 (±0.494)	0.333 (±0.922)	186.200 (±31.196)	6.022 (±0.170)	3.939 (±0.297)
	100	8.100 (±5.695)	4.050 (±2.848)	0.300 (±0.877)	346.867 (±137.375)	6.012 (±0.156)	3.240 (±0.423)
1296	25	18.267 (±3.194)	8.833 (±1.544)	0.367 (±0.928)	165.367 (±44.187)	4.620 (±0.094)	5.805 (±0.025)
	50	17.333 (±5.128)	8.633 (±2.569)	0.333 (±0.922)	136.700 (±44.319)	5.690 (±0.092)	4.575 (±0.161)
	75	10.200 (±1.669)	5.100 (±0.835)	0.333 (±0.922)	192.967 (±36.416)	6.010 (±0.124)	3.976 (±0.153)
	100	6.067 (±4.354)	3.033 (±2.177)	0.133 (±0.507)	365.867 (±119.465)	6.021 (±0.129)	3.109 (±0.460)
1728	25	19.600 (±2.568)	0.000 (±0.000)	0.400 (±0.932)	154.400 (±33.757)	4.641 (±0.078)	5.799 (±0.029)
	50	16.833 (±5.004)	0.000 (±0.000)	0.333 (±0.922)	144.667 (±56.804)	5.695 (±0.118)	4.572 (±0.192)
	75	9.967 (±1.402)	0.000 (±0.000)	0.333 (±0.922)	186.300 (±40.623)	6.014 (±0.126)	3.976 (±0.167)
	100	6.800 (±6.588)	0.000 (±0.000)	0.200 (±0.805)	344.233 (±115.088)	6.019 (±0.128)	3.300 (±0.353)

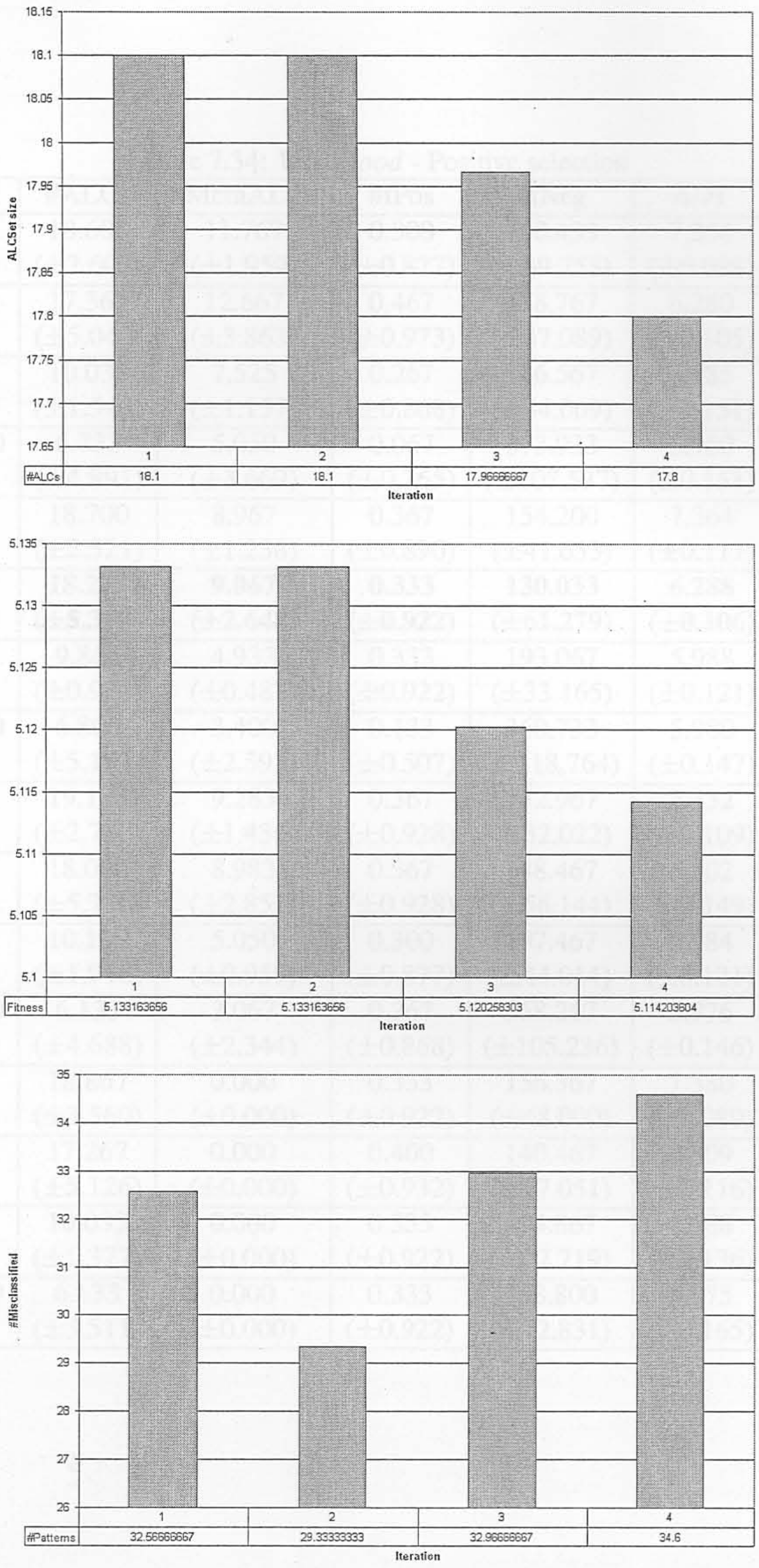


Figure 7.17: *Very Good* - Negative selection with IS=432 and W1=50

Table 7.34: *Very Good* - Positive selection

IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
432	25	18.608 (±2.603)	11.767 (±1.959)	0.300 (±0.877)	150.433 (±39.753)	7.254 (±0.088)	5.696 (±0.062)
	50	17.367 (±5.042)	12.667 (±3.863)	0.467 (±0.973)	138.767 (±47.089)	6.280 (±0.105)	4.524 (±0.176)
	75	10.033 (±1.542)	7.525 (±1.157)	0.267 (±0.868)	186.567 (±34.069)	5.985 (±0.151)	3.970 (±0.218)
	100	6.733 (±4.891)	5.050 (±3.669)	0.067 (±0.365)	373.933 (±107.537)	5.980 (±0.151)	3.202 (±0.386)
864	25	18.700 (±2.521)	8.967 (±1.238)	0.367 (±0.890)	154.200 (±41.633)	7.364 (±0.117)	5.794 (±0.025)
	50	**18.233 (±5.399)**	**9.067 (±2.648)**	**0.333 (±0.922)**	**130.033 (±61.279)**	**6.288 (±0.106)**	**4.553 (±0.181)**
	75	9.867 (±0.973)	4.933 (±0.487)	0.333 (±0.922)	193.067 (±33.165)	5.988 (±0.121)	3.978 (±0.187)
	100	6.800 (±5.182)	3.400 (±2.591)	0.133 (±0.507)	360.733 (±118.764)	5.980 (±0.147)	3.185 (±0.377)
1296	25	19.133 (±2.700)	9.283 (±1.436)	0.367 (±0.928)	152.967 (±32.022)	7.352 (±0.109)	5.792 (±0.023)
	50	18.000 (±5.736)	8.983 (±2.851)	0.367 (±0.928)	148.467 (±56.144)	6.302 (±0.149)	4.566 (±0.195)
	75	10.100 (±1.918)	5.050 (±0.959)	0.300 (±0.877)	197.467 (±44.014)	5.984 (±0.121)	3.970 (±0.150)
	100	6.133 (±4.688)	3.067 (±2.344)	0.267 (±0.868)	368.267 (±105.236)	5.976 (±0.146)	3.161 (±0.514)
1728	25	18.867 (±3.560)	0.000 (±0.000)	0.333 (±0.922)	156.367 (±48.000)	7.380 (±0.089)	5.805 (±0.032)
	50	17.267 (±5.126)	0.000 (±0.000)	0.400 (±0.932)	140.467 (±47.051)	6.309 (±0.116)	4.573 (±0.157)
	75	10.033 (±1.377)	0.000 (±0.000)	0.333 (±0.922)	194.867 (±27.719)	5.988 (±0.136)	3.979 (±0.184)
	100	6.133 (±3.511)	0.000 (±0.000)	0.333 (±0.922)	338.800 (±72.831)	5.975 (±0.165)	3.248 (±0.409)

DATA SET	CLASS AS SELF	GAIS NEGATIVE SELECTION	GAIS POSITIVE SELECTION	C4.5
IRIS	*Setosa*	99.66%	99.62%	99.3%
	Versicolor	97.13%	97.0%	96.0%
	Virginica	94.6%	94.62%	97.3%
WISCONSIN BREAST CANCER	*Benign*	98.907%	98.798%	96.3%
	Malignant	93.395%	93.948%	96.3%
MUSHROOM	*Edible*	88.502%	87.438%	99.9%
	Poisonous	80.383%	79.105%	99.9%
GLASS	*Building-window-float*	92.741%	92.357%	79.3%
	Building-window-nonfloat	83.816%	84.454%	79.7%
	Containers	99.797%	99.797%	95.9%
	Headlamps	99.376%	99.329%	94.9%
	Tableware	99.85%	99.85%	97.7%
	Vehicle-window-float	98.675%	98.8%	91.7%
CAR EVALUATION	*Acceptable*	46.959%	47.968%	95.1%
	Good	86.651%	85.916%	98.7%
	Unacceptable	74.895%	75.017%	99.5%
	Very Good	92.507%	92.455%	100%

Table 7.35: Summarised results

as the self set. The average ADT in the active set of ALCs with *acceptable* as the self set is the lowest with negative selection and the highest with positive selection for all the classes as the self set. These deductions support the bad classification result with *acceptable* as the self set, since the ALCs are widely distributed in problem space (the high average HD) with the lowest space coverage (the low average ADT for negative selection and the high average ADT for positive selection). For *acceptable* or *unacceptable* as the self set the positive selection method had better classification results than the negative selection method and for *good* or *very good* as the self set the negative selection method had better classification results than the positive selection method.

7.6 Comparing the Results

The classification results obtained from GAIS with negative and positive selection is summarised and compared with C4.5 in table 7.35. C4.5 was trained different from GAIS. The experiments with C4.5 also used a 30-fold cross validation training set, but the training set consisted out of self and non-self patterns. The results show that GAIS had on average better classification than C4.5 in classifying the Iris data set, except for *virginica* as self. C4.5 had better classification for the Mushroom data set and the Car evaluation data set, where GAIS had better classification with the Glass data set. C4.5 had better classification with *malignant* as self and GAIS had better classification with *benign* as self in classifying the Wisconsin breast cancer data set. The high misclassification rate of GAIS on the Mushroom and Car Evaluation data sets is due to the low number of ALCs evolved by the GA. The evolved ALCs are widely distributed in space (refer to the high average HD) with a low space coverage (refer to the low average ADT for negative selection and the high average ADT for positive selection). These deductions indicate that better classification results can be obtained when more ALCs are evolved, but with a higher degree of average overlap (lower average HD) among the evolved ALCs. GAIS only evolves an optimal initial set of ALCs. The initial set is kept static during the training process to determine the status of each ALC in the set. Classification performance of GAIS could be improved by replacing the annihilated ALCs with newly evolved ALCs by the GA. When there is overlap among the self patterns and the non-self patterns, the GA needs to evolve a higher number of ALCs to optimally cover the highly distributed non-self space between the self patterns. From these results it can be concluded that depending on the problem that needs to be classified and the selected class as the self set, there are cases that GAIS performs better then C4.5. This deduction supports the *no free lunch theorem* [84].

Chapter 8

Conclusion and Future Work

"It is wise to keep in mind that neither success nor failure is ever final."

- Roger Babson

This chapter concludes the dissertation, discusses the findings and presents ideas relating to possible future work.

8.1 Conclusion

The main objective of this dissertation - to evolve ALCs that have the maximum coverage of non-self space with the least overlap among the ALCs - is addressed. The dissertation started-off with an overview of the functioning of the natural immune system (NIS) - the biological system that protects the body against harmful pathogenic material. The different states of a lymphocyte were also introduced. The dissertation gave an overview of evolutionary computation, focusing on genetic algorithms. A new artificial immune system (namely GAIS) was developed for pattern classification. GAIS uses a genetic algorithm to evolve artificial lymphocytes (ALCs). GAIS evolved the trained ALCs sequentially. Each evolved ALC was added to the set of existing ALCs. The GA was forced with the least overlap restriction to explore different regions in the search space that was not covered by the existing set of ALCs. The evolved ALC was a local optimum in the search space that was not covered by the existing set of ALCs. The ALCs were trained with negative or positive selection to ensure that the ALCs did not detect any of the patterns in the predetermined self set. The active set of evolved ALCs was used to classify patterns. The status of the ALCs was evaluated at predetermined time steps (*IS*) using the life counter threshold function proposed in chapter 5, achieving the sub objective, i.e. proposing a method to dynamically determine the status of an ALC. Annihilated ALCs were removed from the active set of ALCs. Results of GAIS on different data sets were presented in chapter 7 and compared with C4.5. These results showed that there are cases that GAIS performs better then C4.5 and that the overlap among self and non-self patterns influences the number of evolved ALCs and the classification performance of GAIS.

8.2 Future work

In most classification models there are always the danger of overfitting the data during training. Thus for future work, an investigation into the overfitting characteristics of GAIS is necessary to determine if the evolved set of ALCs in GAIS and the proposed life counter threshold function alleviate overfitting of data. An investigation to find the optimal operators in the GA could improve the performance of GAIS as well as the replacement of an annihilated ALC with an evolved ALC. As an unsupervised approach, the proposed life counter threshold function might be used in clustering data to indicate the centroids of the formed clusters. It would also be worthwhile to investigate and analyse the parameters in GAIS to be able to dynamically set these parameters for optimal classification performance and to extent the GAIS algorithm for incremental learning. There are a few observations that need to be more closely examined. Firstly, since the evolved optimal initial set is kept static during the training process, the classification performance of GAIS could be improved by replacing the annihilated ALCs with newly evolved ALCs during training. Lastly, to investigate the influence of *IS* on the classification performance of GAIS, since *IS* determines how frequently the status of an ALC is evaluated. A smaller *IS* results in a more frequent removal of annihilated ALCs from the active set of ALCs. Replacing these annihilated ALCs with newly evolved ALCs, could improve the non-self space coverage and thus the classification performance of GAIS. A comparative study between GAIS, existing AIS classification algorithms and machine learning algorithms is currently being conducted by one of the members in the computational intelligence research group (CIRG).

Bibliography

[1] T. Bäck, H. Schwefel, *An Overview of Evolutionary Algorithms for Parameter Optimization*, Evolutionary Computation, vol. 1, no. 1, pp. 1-23, 1993.

[2] T. Bäck, H. Schwefel, *Evolutionary Computation : An Overview*, Center for Applied Systems Analysis, Dortmund, 1996.

[3] D. Beasley, D.R. Bull, R.R. Martin, *A Sequential Niching Technique for Multimodal Function Optimization*, Evolutionary Computation, vol. 1, no. 2, pp. 101-125, 1993.

[4] D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms : Part I, Fundamentals*, University Computing, 1993.

[5] D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms : Part II, Research Topics*, University Computing, 1993.

[6] C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[7] C. Blake, E. Keogh, C. J. Merz, *UCI - Repository of Machine Learning Databases*, University of California, Irvine, Department of Information and Computer Sciences, http://www.ics.uci.edu/~MLRepository.html, 2002.

[8] M. Bohanec, V. Rajkovic, *Expert System for Decision Making*, Sistemica, vol. 1, no. 1, pp. 145-157, 1990.

[9] R. Brits, A.P. Engelbrecht, F. van den Bergh, *Solving Systems of Unconstrained Equations using Particle Swarm Optimizers*, In Proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 102-107, October 2002.

[10] E. Cantú-Paz, *Order Statistics and Selection Methods of Evolutionary Algorithms*, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, 2002.

[11] J.H. Carter, *The Immune System as a Model for Pattern Recognition and Classification*, Journal of the American Medical Informatics Association, vol. 7, no. 1, pp. 28-41, 2000.

[12] P-C. Chi, *Genetic Search with Proportion Estimates*, In J.D. Schaffer, ed., Proceedings of the Third International Conference on Genetic Algorithms, pp. 92-97, Morgan Kaufmann, 1989.

[13] C.A. Coello Coello, D.A. Van Veldhuizen, G.B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Plenum Pub Corp, 2002.

[14] C. Darwin, *The Origin of Species*, Vol. XI. The Harvard Classics. New York: P.F. Collier & Son, 1909-14, Bartleby.com, 2001.

[15] D. Dasgupta, *Artificial Immune Systems and their Applications*, ed., Berlin: Springer, 1998.

[16] D. Dasgupta, S. Forrest, *An Anomaly Detection Algorithm Inspired by the Immune System*, Chapter 14 in the book entitled Artificial Immune Systems and Their Applications, Publisher: Springer-Verlag, Inc., pp 262-277, January 1999.

[17] L.N. de Castro, J. Timmis, *An Artificial Immune Network for Multimodal Function Optimization*, In Proceedings of IEEE Congress on Evolutionary Computing (CEC'02), pp. 699-674, 2002.

[18] L.N. de Castro, F.J. Von Zuben, *Artificial Immune Systems: Part I - Basic Theory and Applications*, Technical Report - RT DCA 01/99, pp. 95, 1999.

[19] L.N. de Castro, F.J. Von Zuben, *An Evolutionary Immune Network for Data Clustering*, In Proceedings of the IEEE Brazilian Simposium on Artificial Neural Networks (SBRN'00), pp. 84-89, 2000.

[20] L.N. de Castro, F.J. Von Zuben, *Artificial Immune Systems: Part II - A Survey of Applications*, Technical Report - RT DCA 02/00, pp. 65, 2000.

[21] L.N. de Castro, F.J. Von Zuben, *The Clonal Selection Algorithm with Engineering Applications*, In Proceedings of the Genetic and Evolutionary Computational Conference (GECCO'00), Workshop on Artificial Immune Systems and Their Applications, pp. 36-37, 2000.

[22] L.N. de Castro, F.J. Von Zuben, *An Immunological Approach to Initialize Centers of Radial Basis Function Neural Networks*, In Proceedings of the Fifth Brazilian Conference on Neural Networks (CBRN'01), pp. 79-84, 2001.

[23] L.N. de Castro, F.J. Von Zuben, *An Immunological Approach to Initialize Feedforward Neural Network Weights*, In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'01), pp. 126-129, 2001.

[24] L.N. de Castro, F.J. Von Zuben, *Learning and Optimization Using the Clonal Selection Principle*, IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, vol. 6, no. 3, pp. 239-251, 2002.

[25] W. Durham, *Co-Evolution: Genes, Culture, and Human Diversity*, Stanford University Press, Stanford, 1994.

[26] A.P. Engelbrecht, *Computational Intelligence : An Introduction*, Wiley, 2002.

[27] L.J. Fogel, A.J Owens, M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, Wiley, 1966.

[28] S. Forrest, S. Hofmeyr, A. Somayaji, *Computer Immunology (DRAFT)*, Communications of the ACM, vol. 40, no. 10, pp. 88-96, 1997.

[29] S. Forrest, S. Hofmeyr, *Immunology as information processing*, In Design Principles for the Immune System and Other Distributed Autonomous Systems, edited by L.A. Segel and I. Cohen. Santa Fe Institute Studies in the Sciences of Complexity. New York: Oxford University Press, 2001.

[30] S. Forrest, A.S. Perelson, L. Allen, R. Cherukuri, *Self-Nonself Discrimination in a Computer*, In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA: IEEE Computer Society Press, 1994.

[31] J.A. Foster, *Evolutionary Computation*, Department of Computer Science, University of Idaho, February 1998.

[32] A.S. Fraser, *Simulation of Genetic Systems by Automatic Digital Computers*, Australian Journal of Biological Science, vol. 10, pp. 484-491, 1957.

[33] T. Fukuda, K. Mori, M. Tsukiyama, *Parallel search for multi-modal function optimization with diversity and learning of immune algorithm*, In Artificial Immune Systems and Their Applications, 1999.

[34] D.E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.

[35] D.E. Goldberg, J. Richardson, *Genetic Algorithm with Sharing for Multimodal Function Optimization*, In Proceedings of the Second International Conference on Genetic Algorithms, pp. 41-49, 1987.

[36] G.E. Goldberg, K. Deb, *A Comparitive Analysis of Selection Schemes used in Genetic Algorithms*, In G.J.E. Rawlins, ed., Foundations of Genetic Algorithms, pp. 69-93, Morgan Kaufmann, 1991.

[37] F. Gonzalez, D. Dasgupta, R. Kozma, *Combining Negative Selection and Classification Techniques for Anomaly Detection*, In the proceedings of the Congress on Evolutionary Computation, Hawaii, May 12-17, 2002.

[38] R. Hightower, S. Forrest, A.S. Perelson, *The Evolution of Emergent Organization in Immune System Gene Libraries*, In L.J. Eshelman (Ed.) Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA.

[39] S. Hofmeyr, S. Forrest, *Immunity by Design: An Artificial Immune System*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Morgan-Kaufmann, San Francisco, CA, pp. 1289-1296, 1999.

[40] S. Hofmeyr, S. Forrest, *Architecture for an Artificial Immune System*, Evolutionary Computation, vol. 7, no. 1, Morgan-Kaufmann, San Francisco, CA, pp. 1289-1296, 2000.

[41] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[42] J.H. Holland, *ECHO: Explorations of Evolution in a Miniature World*, Proceedings of the Second Conference on Artificial Life, J.D. Farmer, J. Doyne eds., Addison-Wesley, 1990.

[43] J. Horn, *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations.*, PhD thesis, Urbana, University of Illinois, Illinois, Genetic Algorithm Lab, 1997.

[44] J.E. Hunt, D.E. Cooke, *Learning using an Artificial Immune System*, Journal of Network and Computer Applications, vol. 19, pp. 189-212, 1996.

[45] N.K. Jerne, *Towards a network theory of the immune system*, Annals of Immunology (Inst. Pasteur) 125C, pp. 373-389, 1974.

[46] J.H. Jun, D.W. Lee, K.B. Sim, *Realization of cooperative swarm behavior in distributed autonomous robotic systems using artificial immune system*, In Proceedings of IEEE international conference on systems, man and cybernetics held in Tokyo, Japan, vol. 6, pp. 614-619, November 1999.

[47] J. Kennedy, R. C. Eberhart, *Particle Swarm Optimization*, Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, pp. 1942-1948, 1995.

[48] C. Kesmir, R.J. De Boer, *A Mathematical Model on Germinal Center Kinetics and Termination*, Journal of Immunology, vol. 163, no. 5, pp. 2463-2469, 1999.

[49] J. Kim, P.J. Bentley, *Negative Selection and Niching by an Artificial Immune System for Network Intrusion Detection*, Genetic and Evolutionary Computation Conference (GECCO '99), pp. 149-158, Orlando, Florida, July 13-17, 1999.

[50] J. Kim, P.J. Bentley, *Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection*, Genetic and Evolutionary Computation Conference 2001 (GECCO-2001), pp. 1330 - 1337, San Francisco, July 7-11, 2001.

[51] T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, 1995.

[52] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[53] S.W. Mahfoud, *Niching Methods for Genetic Algorithms*, PhD thesis, Genetic Algorithm Lab, University of Illinois, Illinois, IlliGAL Rep. 95001, 1995.

[54] P. Matzinger, *The Danger Model in its Historical Context*, Scandinavian Journal of Immunology, vol. 54, pp. 4-9, 2001.

[55] P. Matzinger, *The Real Function of the Immune System*, http://cmmg.biosci.wayne.edu/asg/polly.html, February 2004.

[56] B.J. Meyer, H.S. Meij, S.V. Grey, A.C. Meyer, *Fisiologie van die mens - Biochemiese, fisiese en fisiologiese begrippe*, Kagiso Tersier, Cape Town, 1996.

[57] B.L. Miller, M.J. Shaw, *Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization*, Technical Report, Genetic Algorithm Lab, Urbana, University of Illinois, Illinois, IlliGAL Rep. 95010, December 1995.

[58] T. Mollestad, *A Rough Set Approach to Data Mining: Extracting a Logic of Default Rules from Data*, PhD Thesis, Department of Computer Science, The Norwegian University of Science and Technology, 1997.

[59] M. Oprea, A.S. Perelson, *Somatic Mutation leads to Efficient Affinity Maturation when Centrocytes recycle back to Centroblasts*, Journal of Immunology, vol. 158, pp. 5155-5162, 1997.

[60] M. Oprea, S. Forrest, *Simulated Evolution of Antibody Gene Libraries under Pathogen Selection*, 1998 IEEE International Conference on Systems, Man and Cybernetics, 1998.

[61] Z. Pawlak, *Rough Sets*, International Journal of Computer and Information Sciences, vol. 11, pp. 341-356, 1982.

[62] A.S. Perelson, *Immune network theory*, Immunological Review, vol. 110, pp. 5-36, 1989.

[63] A.S. Perelson, G. Weisbuch, *Immunology for physicists*, Reviews of Modern Physics, vol. 69, no. 4, October 1997.

[64] M.A. Potter, K.A. De Jong, *The Coevolution of Antibodies for Concept Learning*, In Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V), pp. 530-539, Springer-Verlag, 1998.

[65] S. Pramanik, R. Kozma, D. Dasgupta, *Dynamical Neuro-Representation of an Immune Model and its Application for Data Classification*, In the Proceedings of the International Joint Conference on Neural Networks (IJCNN'02), World Congress on Computational Intelligence (WCCI), 2002.

[66] K.V. Price, *An Introduction to Differential Evolution*, New Ideas in Optimization, D. Corne, M. Dorigo, F. Glover, eds., pp. 79-108, 1999.

[67] I. Rechenberg, *Cybernetic Solution Path of an Experimental Problem*, Royal Aircraft Establishment, Library translation no. 1122, Farnborough, Hants., UK, August 1965.

[68] I. Rechenberg, *Evolution Strategy*, in J.M. Zurada, R. Marks II, C. Robinson, eds., Computational Intelligence - Imitating Life, IEEE Press, pp. 147-159, 1994.

[69] J.T. Richardson, M.R. Palmer, G.E. Liepins, M.R. Hilliard, *Some guidelines for genetic algorithms with penalty functions*, In J.D. Schaffer, ed., Proceedings of the Third International Conference on Genetic Algorithms, pp. 191-197, Morgan Kaufmann, 1989.

[70] S.E. Rouwhorst, A.P. Engelbrecht, *Searching the Forest: Using Decision Trees as Building Blocks for Evolutionary Search in Classification Databases*, The International Conference on Evolutionary Computing, 2000.

[71] L. Schindler, D. Kerrigan, J. Kelly, *Understanding the Immune System - Tutorial*, Science behind the news - National Cancer Institute.

[72] M. Schoenauer, Z Michalewicz, *Evolutionary Computation*, Published in Control and Cybernetics, vol. 26, no. 3, pp 307-338, 1997.

[73] H.-P. Schwefel, *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik*, Diplomarbeit, Technische Universität Berlin, 1965.

[74] A. Somayaji, S. Hofmeyr, S. Forrest, *Principles of a Computer Immune System*, 1997 New Security Paradigms Workshop, pp. 75-82, 1998.

[75] A. Somayaji, S. Forrest, *Automated Response Using System-Call Delays*, Usenix 2000.

[76] J. Timmis, *On Parameter Adjustment of the Immune Inspired Machine Learning Algorithm AINE*, Technical Report 12-00, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent. CT2 7NF., November 2000.

[77] J. Timmis, *Artificial immune systems: A novel data analysis technique inspired by the immune network theory*, Ph.D. thesis, University of Wales, Aberystwyth, 2001.

[78] J. Timmis, M. Neal, *Investigating the Evolution and Stability of a Resource Limited Artificial Immune System*, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 40-41, 2000.

[79] J. Timmis, M. Neal, J. Hunt, *Data Analysis using Artificial Immune Systems, Cluster Analysis and Kohonen Networks: Some comparisons*, In Proceedings of IEEE international conference on systems, man and cybernetics held in Tokyo, Japan, vol.3, pp. 922-927, November 1999.

[80] Y. Watanabe, A. Ishiguro, Y. Uchikawa, *Decentralized Behavior Arbitration Mechanism for Autonomous Mobile Robot using Immune Network*, In Artificial Immune Systems and their Applications, D. Dasgupta, ed., pp. 187-209, Berlin, Springer, 1998.

[81] A. Watkins, L. Boggess, *A New Classifier Based on Resource Limited Artificial Immune Systems*, In Proceedings of Congress on Evolutionary Computation, Part of the 2002 IEEE World Congress on Computational Intelligence held in Honolulu, HI, USA, pp. 1546-1551, May 2002.

[82] A. Watkins, J. Timmis, *Artificial Immune Recognition System (AIRS): Revisions and Refinements*, J. Timmis, P.J. Bentley, eds., 1st International Conference on Artificial Immune Systems, pp. 173-181, University of Kent at Canterbury, September 2002.

[83] S.M. Weiss, C.A. Kulikowski, *Computer Systems that Learn, Classification and Prediction Methods from Statistics, Neural Networks, Machine Learning and Expert Systems*, Morgan Kaufmann, San Mateo, CA, 1991.

[84] D.H. Wolpert, W.G. Macready, *No Free Lunch Theorems for Search*, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1996.

Appendix A

Publications

- A.J. Graaff, A.P. Engelbrecht, *The Artificial Immune System for Fraud Detection in the Telecommunications Environment*, In Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC2003), South Africa, 2003.
- A.J. Graaff, A.P. Engelbrecht, *The Artificial Immune System as Classifier*, South African Institute for Computer Scientists and Information Technologists (SAICSIT2003), In Proceedings of the Post-graduate Research Symposium, pp. 8-9, South Africa, 2003.
- A.J. Graaff, A.P. Engelbrecht, *Using a Threshold Function to Determine the Status of Lymphocytes in the Artificial Immune System*, In Proceedings of the South African Institute for Computer Scientists and Information Technologists (SAICSIT2003), pp. 268-274, South Africa, 2003. A revised version is also submitted to the South African Computer Journal (SACJ).

Appendix B

Symbols

Tables B.1 and B.2 list and define the symbols used throughout this dissertation.

Table B.1: Table of symbols

Symbol	Meaning
a_{neg}	Maximum ADT for an ALC trained with negative selection
a_{pos}	Minimum ADT for an ALC trained with positive selection
b	The number of bins or groups
c	Attribute counter in a pattern
d_l	l-th ALC in D
d_{l_i}	Bit i in receptor of d_l
e	The rate of elitism in the GA
$f(\lambda_{Hyper}, x_{Hyper})$	Hyperbolic Tangent function where λ_{Hyper} controls the steepness of $f(\lambda_{Hyper}, x_{Hyper})$
g	Generation counter in the GA
$g(\lambda_{Sigmoid}, x_{Sigmoid})$	Sigmoid function where $\lambda_{Sigmoid}$ controls the steepness of $g(\lambda_{Sigmoid}, x_{Sigmoid})$
k	Length of an ALC's receptor

Table B.2: Table of symbols (continued)

Symbol	Meaning
$max_{c=1,\ldots C}\{x_{c,j}\}$	Maximum value for attribute c in a data set
$min_{c=1,\ldots C}\{x_{c,j}\}$	Minimum value for attribute c in a data set
p_c	Crossover probability
p_m	Mutation probability
r_i	Bit i in receptor $\mathbf{r}$
w_1, w_2	The weights that influence the objectives in the fitness function
$x_{c,j}$	Value of attribute c of pattern j in a data set
$y = \|A_i - B_i\|$	Returns the absolute difference between the values of gene i in A and B
A_i	Gene i in parent A
B_i	Gene i in parent B
C	Number of attributes in a pattern
D	Active set of ALCs
D_l	ALC set D after adding l-th evolved ALC
D_L	Initial active ALC set with L evolved ALCs.
G	The maximum number of generations in the GA
$G(x_{c,j})$	Assigns an attribute's value to the correct bin or group
H_g	Population in GA at generation g
I	The number of ALCs in the GA population
K	Population size in the GA
$Max(A_i, B_i)$	Returns the chromosome with the highest value in gene i
$Min(A_i, B_i)$	Returns the chromosome with the lowest value in gene i
$N = \frac{log(b)}{log2}$	Calculates the number of bits necessary to represent b bins or groups
O_i	Gene i in offspring O
P	The number of ALCs in D
$WindowSize$	The number of generations to calculate the moving average on a population's fitness
$XOR(x_i, r_i)$	Exclusive-or between the i-th bit in $\mathbf{x}$ and $\mathbf{r}$
β	Minimum matching ratio of an ALC(space coverage)
$\gamma(\mathbf{x}, \mathbf{r})$	Hamming distance between $\mathbf{x}$ and $\mathbf{r}$
μ_g	The moving average at generation g in the GA
μ_T	The threshold for the difference in moving averages in the GA
$\mu(D_l)$	Average fitness of ALC set D_l
$\mu(H_g)$	The average fitness of population H_g in the GA
$\xi \sim U(0,1)$	Random number generated from a Uniform distribution
ς	Generation gap
τ	The life counter threshold function
υ_{neg}	Negative selection fitness function
υ_{pos}	Positive selection fitness function
$\chi(D, \mathbf{r})$	Average hamming distance between $\mathbf{r}$ and the set D
$\mho$	Finite search space

Appendix C

Abbreviations

ADT:	Affinity Distance Threshold
AIS:	Artificial Immune System
ALC:	Artificial Lymphocyte
EA:	Evolutionary Algorithm
EC:	Evolutionary Computation
EMR:	Expected Matching Ratio
EP:	Evolutionary Programming
ES:	Evolutionary Strategies
GA:	Genetic Algorithm
GP:	Genetic Programming
HC:	Hit Counter
HD:	Hamming Distance

HR:	Hit Ratio
HTC:	Helper-T-Cell
IS:	Iteration Size
LC:	Life Counter
MHC:	Major Histocompatibility Complex
NIS:	Natural Immune System
NKTC:	Natural-Killer-T-Cell

Appendix D

Glossary

Affinity: A force that causes the HTC to elect an MHC on the surface of the B-Cell with which the HTC has a stronger binding to unite, rather than with another MHC with a weaker binding.

Affinity Distance Threshold: The receptor on the lymphocyte must bind with a certain affinity to the antigen for the antigen to be detected. This affinity is referred to as the affinity distance threshold.

Alleles: Specific values from the domain of the corresponding parameter assigned to the gene.

Annihilated: The status of a frequently suppressed lymphocyte.

Antibody: Cell produced by a B-Cell after detecting antigen.

Antigen: Material that can trigger an immune response. Antigens can be either bacteria, fungi, parasites and/or viruses.

B-Cell: A lymphocyte that produces antibodies after detecting antigen.

Chromosome: A potential solution to the problem that needs to be optimised.

Clone: Exact copy of a stem cell.

Crossover: Process of exchanging genetic material between two or more selected chromosomes to produce offspring.

Cytokines: Encourages cell growth, promote cell activation or destroy target cells.

Donor cells: Transplanted blood cells obtained through transplanted organs or blood.

Epitope: The small segments on the surface of an antigen. Epitopes trigger a specific immune response and antibodies bind to these epitopes.

Expected Matching Ratio: The detection ratio of non-self patterns expected from an artificial lymphocyte.

Fitness: Maps the chromosome's representation into a scalar value. Quantifies the quality of the chromosome. The scalar value of the chromosome indicates how close a chromosome is to the optimal solution.

Gene: A parameter in the chromosome's representation with values in the same space as the function being optimised (referred to as *phenotype-space*).

Generation: A population of individuals at a specified time step, evolved from previous populations at earlier time steps.

Genotype-space: An intermediate space to which parameter-values in the chromosome are mapped to.

Helper-T-Cell: Strengthens the cloning of a B-Cell into a plasma cell.

Hit Counter: Keeps record of the number of matched non-self patterns by an artificial lymphocyte.

Hit Ratio: Calculates the ratio at which an artificial lymphocyte matched non-self patterns.

Immature: A T-Cell that is not self-tolerant.

Immune response: The body's reaction to antigens so that the antigens are eliminated to prevent damage to the body.

Iteration Size: The specified number of incoming patterns before calculating the life counter of an artificial lymphocyte.

Life counter: A threshold function that determines in which state an artificial lymphocyte is in its life cycle.

Lymphocyte: Detects any antigens in the body.

Lymphoid organ: Responsible for the growth, development and deployment of the lymphocytes in the immune system.

Lymphokines: Known as cytokines.

Macrophages: Are versatile cells that secrete powerful chemicals and plays an important role in T-Cell activation.

Major Histocompatibility Complex: These molecules are on the surface of a cell and their main function is to bring to light the internal structure of a cell.

Mature: A T-Cell that does not have receptors that bind with molecules that represent *self* cells.

Memory: A B-Cell that frequently detects non-self cells. The function of memory cells is to proliferate to plasma cells for a faster reaction to frequently encountered antigens and produce antibodies for the antigens.

Minimum matching ratio: The coverage of non-self space by an artificial lymphocyte.

Monocytes: Type of phagocyte.

Monokines: Known as cytokines.

Monomeric receptor: A chemical compound that can undergo a chemical reaction with other molecules to form larger molecules.

Multi-modal: Some functions have multiple solutions and are known as multi-modal functions.

Mutation: Randomly changing the genetic representation of a chromosome.

Natural-Killer-T-Cell: Binds to the Major Histocompatibility Complex-molecule and destroys the virally infected cell.

Negative selection: Specific selection method to train artificial lymphocytes to become self-tolerant. The trained ALC covers the non-self space.

Neutrophils: Type of phagocyte.

Niching: A technique that has been developed to find multiple solutions in multi-modal functions.

Non-self: Unwanted foreign cells that is harmful to the body.

Pathogen: Foreign body material (referred to as antigen).

Peptides: The partitions of an antigen.

Phagocytes: Cells that are large cell- and particle-devouring white cells.

Phenotype-space: The domain of the fitness function.

Plasma cell: A B-Cell that produces antibodies.

Positive selection: Specific selection method to train artificial lymphocytes to become self-tolerant. The trained ALC covers the self space.

Self: Normal functioning cells in the body.

T-Cell: A lymphocyte that becomes mature in the thymus. There are two types of T-Cell: Helper-T-Cell and Natural-Killer-T-Cell.