

# Optimised Coverage of Non-self with Evolved Lymphocytes in an Artificial Immune System

A.J. Graaff<sup>1</sup> and A.P. Engelbrecht<sup>2</sup>

<sup>1</sup>Department of Computer Science, Computational Intelligence Research Group (CIRG),  
School of Information Technology,  
University of Pretoria, Pretoria 0002, South Africa.  
*agraaff@cs.up.ac.za*

<sup>2</sup>Department of Computer Science, Computational Intelligence Research Group (CIRG),  
School of Information Technology,  
University of Pretoria, Pretoria 0002, South Africa.  
*engel@cs.up.ac.za*

**Abstract:** The natural immune system (NIS) protects the body against unwanted foreign material (non-self cells) that could damage the body (self cells). The NIS can be modeled into an artificial immune system (AIS) to detect any non-self patterns in a non-biological environment. Detectors in the NIS can change from their initial mature status to memory status detectors or to annihilated status. A memory detector is a detector that frequently detects non-self cells and is a general detector for a subset of non-self cells. The NIS uses these memory detectors in a faster response to non-self cells. The purpose of this paper is to present the genetic artificial immune system (GAIS) which evolves these non-self detectors and determine their state using a life counter function. Only detectors with mature or memory status are used to detect non-self. Thus, the number of detectors is dynamically determined by the life counter function. In the paper GAIS is applied to different classification problems.

**Keywords:** artificial lymphocytes, non-self, negative selection, memory, classification

## I. Introduction

Artificial intelligence consists of many paradigms, including artificial neural networks (ANNs) [4], evolutionary computation (EC) [1], swarm intelligence (SI) [36, 20], artificial immune systems (AISs) [9] and fuzzy systems [60]. The performance of algorithms within these paradigms differ among different problem domains. A problem can be solved by a number of different algorithms. It is therefore important to have a good definition of the problem so that the most appropriate algorithm is selected to solve the problem. Some of these models are specifically focused at solving classification problems [7, 27]. Classification is the process

of finding classification boundaries to separate patterns that belong to different classes. Some of the above mentioned paradigms are based on natural processes. Natural processes that have been modeled to develop classification techniques include ANNs that model biological neural networks [4], EC techniques that model the natural evolution of organisms [1], SI that models the behavior of a structured collection of interacting organisms to solve a global objective [36] and AISs that model the functionality and principles of the natural immune system to adapt to changing environments [13, 53]. These algorithms are referred to as computational intelligence (CI) techniques while algorithms like decision trees form part of the classical machine learning (ML) methods.

Both methods from classical ML and CI can be grouped into supervised and unsupervised learning methods. In supervised learning a model is trained with pre-classified negative and positive patterns that serve as examples for the model. These example patterns are pre-classified according to a certain concept or rule. Unsupervised learning on the other hand is based on a process of automatically discovering similar patterns in the data without guidance from an external teacher [40]. Methods from these ML and CI paradigms require sufficient training sets consisting of negative and positive examples of classes.

The artificial immune system (AIS) is a relatively new paradigm of algorithms that have been applied to classification problems [10, 39, 57]. An AIS models the natural immune system (NIS) to detect or classify foreign patterns in a non-biological environment. The NIS does not only have the ability to learn valid patterns and to detect or classify foreign patterns, but also has a memory. The NIS

is capable of memorising general foreign pattern structures [43, 47]. A unique characteristic of the AIS is that training requires only positive examples. After training, the AIS has the ability to classify foreign (negative) patterns from the positive patterns. This makes the AIS an ideal candidate for classification problems where only one class of patterns is available for training.

The AIS presented in this paper makes use of a set of artificial lymphocytes (ALCs) to detect negative patterns. The set is populated by mature ALCs, which are evolved from a genetic algorithm (GA). The GA has as its main objective to evolve ALCs with maximum space coverage of possible negative patterns and least overlap among ALCs. After each evolved ALC is added to the set, the GA is forced to explore different regions in the search space that is not yet covered by the mature ALCs in the set. An ALC evolves to maturity after it has been trained on a set of positive patterns in the GA. There are two methods to train an ALC, positive selection and negative selection [23]. In addition to the mature status of an ALC, there is also the following states: Immature, Memory and Annihilated. An ALC with immature status has not yet been trained by a GA. Memory ALCs are those ALCs that frequently detect negative patterns and annihilated ALCs seldom detect any negative patterns and need to be replaced in the set of ALCs. An optimal set of ALCs can be defined as a set that consists of ALCs that frequently detect negative patterns. It is therefore important to distinguish between ALCs with high probability to detect negative patterns from those ALCs with poor detection, i.e. distinguishing ALCs based on status.

The purpose of this paper is to illustrate how a genetic algorithm can be used to evolve detectors that are self-tolerant and how to determine if a detector has a high probability to detect non-self patterns, using a threshold function as presented by Graaff and Engelbrecht [28]. The paper also presents as a sub-objective a life counter function [28] as a model for automatic transformation between states of an ALC. The number of ALCs in the proposed model is thus dynamically determined by the life counter function based on the ALCs status, since only memory and mature ALCs are used to detect non-self patterns. The outline of the paper is as follows: A brief overview on the natural immune system is given in section II, followed by existing AIS models and applications in section III. Sections IV to VI present an overview of GAIS (genetic artificial immune system) with the life counter function, and section VII discusses the results obtained from GAIS on different classification problems. Section VIII concludes this paper and discusses future work in AISs.

## II. A Brief Overview of the Natural Immune System

The human body has many different mechanisms to defend itself against pathogenic material. These defence mechanisms are among others the skin that covers the body and the natural immune system. The natural immune system (NIS) detects foreign material inside the body that could be harmful to the body. The NIS differentiates between normal cells (self cells) and foreign cells (non-self cells). The immune system works on the principle of a pattern recognition system, recognising unwanted patterns (non-self cells) from the normal patterns (self cells) [47].

### A. Views on Non-self

To date there are two different views on non-self cells. The classical view of the NIS is that all non-self is harmful to the body and needs to be detected and destroyed by the NIS [43]. A different theory on non-self was introduced by Matzinger, known as danger theory [41, 42]. Danger theory does not classify all foreign or non-self cells as dangerous to the body, i.e. non-self cells are further classified into dangerous and non-dangerous. According to danger theory, the NIS only reacts to dangerous non-self cells. Focusing on the classical view of the NIS, the remainder of this section explains and discusses the different parts of the NIS which are crucial to the detection and annihilation of non-self cells.

### B. The Lymphocytes

The natural immune system mostly consists of lymphocytes and lymphoid organs. These organs are mainly the thymus and bone marrow. The bone marrow is responsible for creating new lymphocytes in the immune system. The lymphocytes are used to detect any foreign or unwanted cells in the body. These foreign cells are known as antigens [50].

There are two types of lymphocytes, namely the T-Cell and B-Cell. Both of these lymphocytes have receptor molecules on their surfaces that bind to other cells. The receptor of the T-Cell binds to molecules that are on the surface of other cells. The molecules on the surface of cells are named Major Histocompatibility Complex molecules (MHC-molecules). The MHC-molecules bring to light the internal structure of a cell and can be differentiated into Type I and Type II MHCs. MHC-molecules of Type I is on the surface of any cell and MHC-molecules of Type II mainly on B-Cells [47].

Binding of a T-Cell's receptor to an MHC can only occur if the T-Cell is mature. A T-Cell becomes mature (in the thymus) if and only if it does not have receptors that will bind with molecules that represent self cells. It is therefore very important that the T-Cell can differentiate between self and non-self cells [47].

There are two types of T-Cells: The Helper-T-Cell (HTC) and the Natural-Killer-T-Cell (NKTC). Co-operation between the different lymphocytes to detect an antigen is explained next.

### C. Detecting the Antigen

Different from T-Cells, B-Cells leave the bone marrow as mature lymphocytes. B-Cells collect antigen in the body and partition the collected antigen into peptides, which in turn are brought to the surfaces of the B-Cells by MHCs of Type II (as illustrated in Figure 1). The receptor of an HTC then binds to an MHC on a B-Cell and secretes lymphokines [50], which proliferate or suppress the B-Cell's response to the antigen. This response is known as the primary response. If the HTC binds to the MHC with a high affinity, the B-Cell is proliferated. A proliferated B-Cell produces antibodies with the same structure as represented by the peptides. A B-Cell becomes proliferated when the B-Cell has grown into a clone, which can produce antibodies [43]. A B-Cell can proliferate into one of two types of cells: plasma cells and memory cells. The function of memory cells is to proliferate to plasma cells for a faster response to frequently encountered antigens and produce antibodies for the antigens. A plasma cell is a B-Cell that produces antibodies. Antibodies form part of self and bind to antigen to de-activate the antigen [43].

When the HTC does not bind with a high affinity, the response of the B-Cell is suppressed. A frequently suppressed B-Cell becomes annihilated and needs to be replaced by a newly created B-Cell. B-Cells with memory status are used by the immune system in a secondary response to frequently seen antigen with the same structure. The secondary response is faster than the above-mentioned primary response, since there is no binding necessary between the HTC and a memory B-Cell to be able to produce antibodies [47].

The Natural-Killer-T-Cell (NKTC) binds only to MHCs of type I. Type I MHCs are found on all cells and they bring to light any viral proteins from a virally infected cell. The NKTC binds to the MHC of a virally infected cell and destroys itself with the infected cell [47].

## III. The Artificial Immune System

Research done on the functioning of the natural immune system (NIS) has led to different theories and models. Although the functioning of the NIS is not yet fully understood, some of these theories have valid arguments. The most familiar of these theories are the classical view of the NIS (as explained in section II), danger theory [41, 42] and network theory [34, 46]. There are a few common characteristics between these theories namely,

- the NIS has the capability to learn the structure of normal cells;
- the NIS has the ability to memorise the structure of frequently encountered foreign cells;
- the memory of the NIS is used in a faster secondary response to foreign cells with identical or similar structure;
- the NIS functions as a pattern recognition and classification system; and
- the NIS is a complex, distributive system.

These characteristics of the NIS inspired the modeling of the NIS into an artificial immune system (AIS) for application in non-biological environments mainly, intrusion detection [10, 23, 27, 38, 39]. Therefore, the AIS mimics the models of the natural immune system's functions and principles. The AIS only needs to be trained on one class of positive patterns (self) to detect or classify negative patterns of different classes (non-self). This advantage makes the AIS unique from other computational models. There is, however, a drawback to this advantage: A limited knowledge on positive patterns, or an incomplete representation of positive examples, can lead to misclassification of non-self patterns. Existing artificial immune system models are discussed in section III(A). A new approach which is taken for the proposed genetic artificial immune system (GAIS), is presented in section III(B). Some applications of AISs are summarised in section III(C).

### A. Natural to Artificial

The NIS is a very complex system. Therefore, modeling of the NIS into an AIS requires a clear understanding of the structure and functioning of the NIS. 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 secretion of antibodies is dependant on the affinity with which a T-Cell binds to the B-Cell's MHC. Based on the classical view of the NIS (as discussed in section II), T-cells undergo a maturation process in the thymus to become self-tolerant. Forrest developed the negative selection method for training these T-Cell detectors in an AIS [23]. Negative selection monitors randomly created T-Cells and replaces a T-Cell that detects any self pattern, i.e. T-Cells which are trained with negative selection are self-tolerant. A drawback of negative selection is that changes in self and non-self result in re-training of T-Cell detectors. It is computationally inefficient to generate self-tolerant detectors to cover all of non-self space, while only some of non-self is harmful and only some of these detectors will actually detect non-self. Positive selection does not solve this problem either. Positive selection monitors randomly created T-Cells and replaces a T-Cell that does not detect all of the self patterns, i.e. T-Cells which

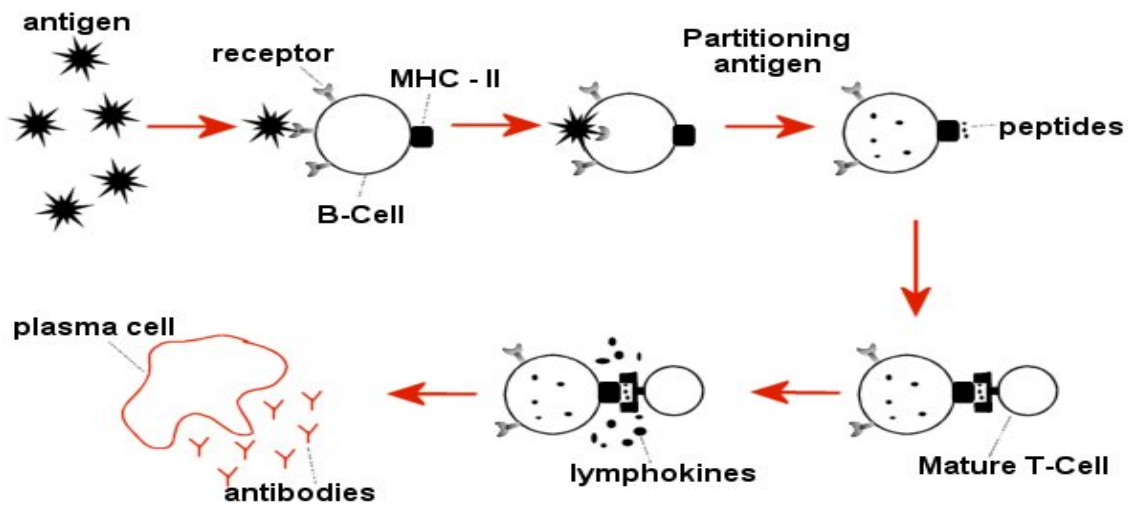


Figure. 1: Detection of antigen by B-Cell

are trained with positive selection are non-self-tolerant. It could also be the case that there is some overlap between self and non-self space. Matzinger presented danger theory as an alternative to the classical view [41, 42]. The main difference between danger theory and the classical view is the self-non-self discrimination. Danger theory does not respond to everything that is non-self, instead it concentrates more on what is dangerous rather than what is non-self. Thus, danger theory discriminates between dangerous and non-dangerous of which both self and non-self can form part of. The network theory of Jerne states that the B-Cells are interconnected to form a network of cells [34, 46]. The B-Cells in the network respond to foreign cells. An activated B-Cell stimulates all the other B-Cells to which it is connected in the network. A B-Cell becomes activated if it detects a foreign cell. Work that has been done in AISs on network theory of B-Cells can be found in [53, 55, 56].

Focusing on the classical view of the NIS, there are a few basic concepts that must be considered to model the proposed artificial immune system:

- There are trained detectors (artificial lymphocytes) that detect non-self patterns with a certain affinity.
- 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 needs a good repository of self patterns or self and non-self patterns to train the artificial lymphocytes (ALCs) to be self-tolerant.

- 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 [22, 30, 31] similar architectures for an AIS are proposed and used for security. Some of the existing AIS models based on the above listed concepts, will be discussed next.

The first three concepts above highlight the affinity measurement between an ALC and a pattern as well as the structure of the ALC. There are different approaches to represent patterns in an AIS and to measure the affinity between these patterns and the ALCs. The measured affinity between a pattern and an ALC needs to exceed a certain threshold for the pattern to be detected by the ALC. Patterns can be represented as vectors of floating-point values [55], nominal-valued vectors [23], or binary strings [15, 33]. Depending on the representation of the patterns in the AIS, the Euclidean distance [55] or the Hamming distance [15] between an ALC and a pattern can be used as a measure of affinity. The above-mentioned measurements indicate the similarity between an ALC and the pattern. A low similarity value indicates a stronger affinity between an ALC and the pattern. The measurement of affinity can also be done using the  $r$ -continuous matching rule [23, 33], which 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.

Training an AIS to find an optimal set of detectors can be done supervised or unsupervised. In supervised learning, the training set consists of self patterns [23], non-self patterns [29, 45], 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. The ALCs can either be randomly initialised [23] or evolved with a genetic algorithm to have a larger detection ratio of non-self patterns in the training set [29]. Forrest *et al.* [23] 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 an 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 in self patterns, thus keeping the ALC set self-tolerant. The negative selection method can also be used to train ALCs on continuous-valued self patterns [27]. The continuous-valued negative selection method evolves ALCs that are the furthest away from the training set of self patterns with the least overlap among the ALCs 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 a non-self pattern in the testing set of patterns. Thus, the trained ALC might never detect any of the non-self patterns in the testing set. Another approach is to use an evolutionary process to evolve ALCs towards non-self and to maintain diversity among the ALCs [38]. The AIS in [48] applies a co-evolutionary genetic algorithm to evolve ALCs. The training set in [48] consists of self and non-self patterns. ALCs are evolved towards the non-self patterns and away from the self patterns in the training set. 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.

The above models are supervised training methods. CLONALG, developed by de Castro [15, 18], is an unsupervised approach that models the clonal selection process of the natural immune system. CLONALG performs machine-learning and pattern recognition tasks. In CLONALG the patterns and ALCs are presented as vectors of floating-point values. The euclidean distance between an ALC and a pattern is used to measure the affinity. A low euclidean distance value indicates a stronger affinity between an ALC and a pattern. CLONALG perceives all the patterns in the training set as non-self patterns. CLONALG uses a set of ALCs with a fixed size of  $S$ . Initially all ALCs are randomly initialised in the set. Each pattern is then randomly selected from the training set (without replacement) and presented to

the set of ALCs. The affinity between the selected pattern and each ALC in the set is calculated. The ALCs with a higher affinity value than a predetermined affinity threshold, are cloned. The clones are mutated and the affinity between the selected training pattern and each mutated clone is calculated. The mutated clones are then added to the set of ALCs and the set is sorted in decreasing order of affinity. The first  $S$  ALCs in the sorted set are then selected as the new set of ALCs. A selection of  $M$  ALCs with high affinity goes into a memory pool. A number of randomly initialised ALCs replaces the ALCs with low affinity in the new set. The next pattern is then selected from the training set and presented to the new set of ALCs until all patterns have been selected. The algorithm can also be used to solve complex problems, eg. multi-modal function optimisation [15, 18].

As mentioned above, one of the drawbacks of negative selection is that it is computationally inefficient to generate self-tolerant detectors to cover all of non-self space. The concept of artificial recognition balls (ARBs) was introduced by Timmis [54] for a resource limited AIS. In this model there is a direct mapping between an ALC and a resource, i.e. an ALC is seen as a resource. An ARB has a general representation for a number of ALCs. Each ARB allocates a number of resources based on its stimulation level. The stimulation level of an ARB is directly proportional to the affinity between an ARB and a pattern. The affinity between an ARB and a pattern is measured by the euclidean distance between the ARB and the pattern. Thus, the lower the euclidean distance value, the stronger the affinity which implies a higher stimulation level. The total number of resources of the system is bounded. Watkins adopted the concept of ARBs [57] for a new classification model. To be able to identify a memory cell, a training set of patterns is presented to the system. The system perceives all the patterns in the training set as non-self. A subset of the training set is used to create an initial batch of memory cells. Each of the remaining non-self training patterns is 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 number of resources for each ARB increases through cloning, 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. 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.

Timmis implemented the network theory by interconnecting all of the ARBs in an AIS [53]. Links are established between ARBs with a high affinity between them. Therefore a network of linked ARBs is formed based on the affinity among the ARBs. In the network of linked ARBs, clusters are formed by ARBs based on their high affinity between each other and the presented patterns. These clusters represent clusters in the data set. The ARB-network model of Timmis resulted in a successful unsupervised training method to visualise data. Another unsupervised approach to cluster data, developed by de Castro and Von Zuben [13], constructs an edge-weighted graph of B-Cells. This model is referred to as aiNet. Some of the B-Cells are connected with edges, with a weight assigned to each edge. The assigned weight represents the connection strength between the connected B-Cells. 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, thus increasing the weight between the connected B-Cells. Immune network theory has also been successfully developed and applied to optimise multi-modal functions [11].

### B. The Genetic Artificial Immune System

A high-level overview of the Genetic Artificial Immune System is illustrated in Figure 2. The Genetic Artificial Immune System (GAIS) presented in this paper represents all patterns in space as binary vectors and uses the hamming distance as affinity measure. A GA is used to evolve a set of ALCs with maximum non-self space coverage and minimum overlap among existing ALCs (explained in section V). The ALCs can either be trained with an adapted negative selection method (explained in section IV(B.1)), or with an adapted positive selection method (explained in section IV(B.2)). The affinity threshold of an ALC is used to determine a match with a non-self pattern. With the adapted negative selection method the affinity threshold is determined by the distance to the closest self pattern from the ALC. The algorithm is supervised, using 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 [38]), 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.* [29] and Oprea [45]). The main goal is 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. Therefore, a proposed transition function, the life counter function (explained

in section IV(D)), determines an ALC's status (defined in section IV(A)). ALCs with annihilated status are removed in an attempt to have only mature and memory ALCs with optimum classification of non-self patterns. Thus, the number of ALCs in GAIS is dynamically determined by the life counter function based on the ALCs status, since only memory and mature ALCs are used to detect non-self patterns. GAIS has the advantage to classify patterns in a problem space where only positive patterns are available for training.

### C. Applications of the Artificial Immune System

The artificial immune system has been successfully applied to many problem domains. These domains range from network intrusion and anomaly detection [10, 21, 22, 27, 30, 31, 38, 39, 51, 52] to data classification models [49, 58] (the model of Pramanik *et al.* [49] bridges the models of [44] and [37]), virus detection [23], concept learning [48], data clustering [13], robotics [35, 56], pattern recognition and data mining [7, 33, 53, 55]. The AIS has also been applied to the initialisation of feed-forward neural network weights [17], the initialisation of centers of a radial basis function neural network [16] and the optimisation of multi-modal functions [11, 25]. The interested reader is referred to [9, 12, 14] for more information on AIS applications.

## IV. The Artificial Lymphocyte in GAIS

Artificial lymphocytes (ALCs) are used by the AIS to detect and classify non-self patterns from self patterns. It is assumed that all patterns in the AIS are represented by bit-strings. The different lymphocytes (as explained in section II(A)) are modeled into a general ALC with a bit-string receptor. The receptor has a length  $k$ , equal to the length of a self and non-self pattern. The ALC must adhere to an affinity-distance threshold (ADT) to be able to detect a non-self pattern in a non-biological environment. From the previous requirement for non-self detection it can be derived that the ALC's receptor cannot bind or match a non-self pattern with an affinity that is greater than the size of the pattern. Therefore,  $ADT \leq k$ .

Only mature ALCs are used to detect non-self patterns. An ALC has mature status if and only if the ALC's receptor does not overlap with any of the known self patterns, used for training. The receptor of an ALC is randomly initialised. Training is then done by measuring the receptor of an ALC against the static incomplete set of known self patterns to determine the ALC's maximum affinity-distance threshold, as discussed in section IV(B). Calculating the maximum ADT will prevent the ALC to overlap with the known self patterns in the training set and therefore prevent the ALC to detect any of the known self patterns.

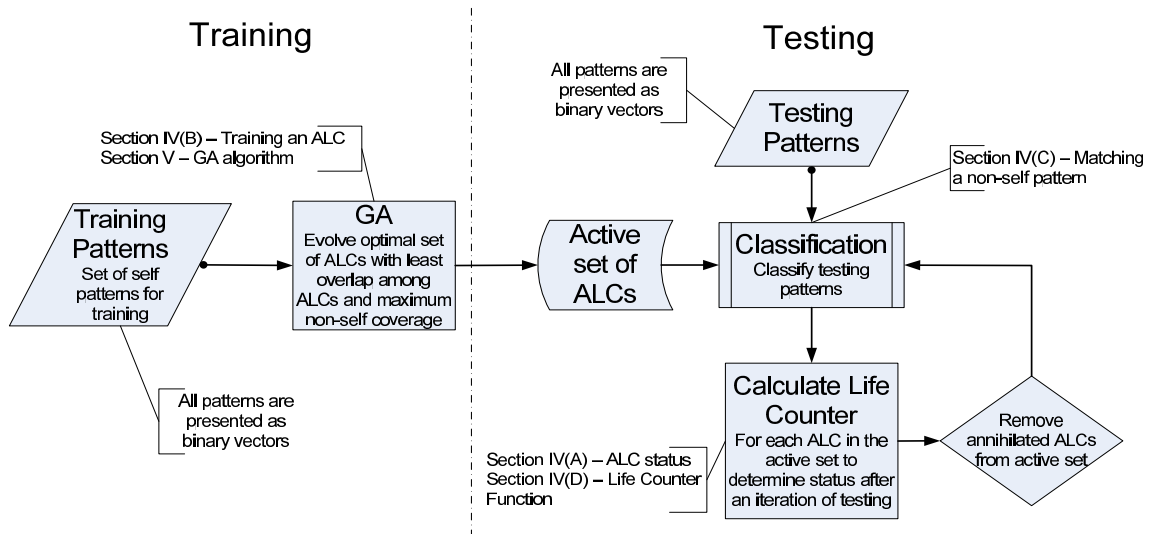


Figure. 2: High-level overview of GAIS

An ALC’s status is determined by a threshold function called the *Life Counter (LC)*, which was presented and explained in [28]. The AIS uses the status of an ALC to determine which of the ALCs need to be discarded and replaced in the set of ALCs. The status is also an indication of an ALC’s priority among the other ALCs in the set.

A. *Life Cycle of the Artificial Lymphocyte*

An ALC can have different states in its life cycle. Figure 3 shows that the status of a trained ALC can be one of four states, namely memory, mature, immature or annihilated. The status of an ALC determines its priority in a group of ALCs: states are prioritised into high, medium and low. The life counter maps the states to a continuous range (0, 1) (i.e.  $LC : \{low, medium, high\} \rightarrow (0, 1)$ ). 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 [28]. The different states are explained below.

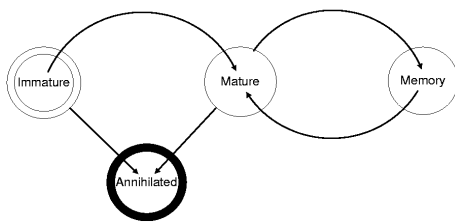


Figure. 3: Life cycle of a lymphocyte

1) *Immature (no priority)*

An immature-ALC has not yet been trained on non-self patterns and is therefore not self-tolerant. The immature status is the initial status of randomly initialised ALCs and these ALCs have no LC-value. [28].

2) *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. It is the annihilated ALCs that must be replaced by newly constructed or evolved ALCs. An ALC with annihilated status have an LC-value which is close to 0.0 [28].

3) *Mature (medium priority)*

A mature-ALC detects non-self patterns on a regular basis. An ALC in mature status will be demoted to annihilated status if the ALC starts to detect less or none of the non-self patterns. If the ALC starts to detect non-self patterns more frequently it will be promoted to memory status [28].

4) *Memory (high priority)*

Memory-ALCs detect non-self patterns more frequently than mature-ALCs. Memory-ALCs are given a higher priority than other ALCs by evaluating an incoming pattern 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 memory status have an LC-value which is more biased to 1.0 [28].

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 the annihilated status (low priority) have a higher probability to be replaced by newly constructed ALCs than ALCs with mature status (medium priority). As long as an ALC has memory status (high priority), the ALC will never be replaced with a newly constructed ALC, since the memory ALC has a higher probability in classifying a non-self 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 dynamically determine the status of an ALC and to automatically change the status of an ALC if required [28].

### B. Training an ALC to Cover Non-self Space

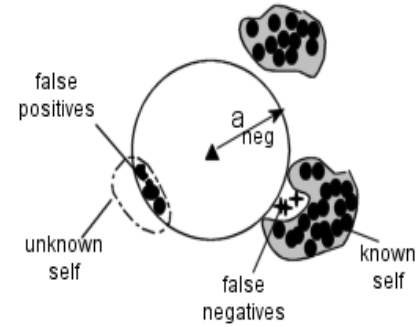
An ALC can be trained with either the positive selection method or the negative selection method. An ALC is trained with an incomplete static set of self patterns. A non-self pattern can be incorrectly classified by an ALC as a self pattern, which is referred to as a *false negative*. A self pattern can also be incorrectly classified by an ALC as a non-self pattern, referred to as a *false positive*. Prior to training, the bit-string receptor of an ALC is randomly initialised. The two training methods differ in that negative selection trains an ALC not to match any self pattern in the training set, in contrast to positive selection, where an ALC is trained to match all of the self patterns in the training set. Both negative and positive selection methods determine the best ADT for the ALC. Adapted versions of the two selection methods are described next.

#### 1) Adapted Negative Selection

The adapted negative selection method trains an ALC to have a maximum affinity-distance threshold,  $a_{neg}$ , that does not overlap with the self set (as illustrated in Figure 4). A pattern will be detected as non-self by an ALC if the hamming distance between the pattern and the ALC is less than  $a_{neg}$ . To guarantee a maximum  $a_{neg}$  with no overlap with self,  $a_{neg}$  is set to the hamming distance of the nearest self pattern to the ALC. Figure 4 shows that, with the adapted negative selection, the self pattern with the smallest hamming distance to the ALC is used to determine the maximum  $a_{neg}$ .

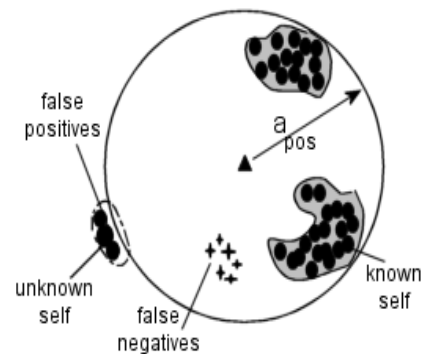
#### 2) Adapted Positive Selection

The adapted positive selection method trains an ALC to have a minimum affinity-distance threshold,  $a_{pos}$ , that does overlap with the self set (as illustrated in Figure 5). A pattern will be detected as non-self if the hamming distance between the pattern and the ALC is greater than  $a_{pos}$ . To guarantee a minimum  $a_{pos}$  with overlap with self,  $a_{pos}$  is set to the hamming



**Figure 4:** Venn-diagram of Adapted Negative Selection ALC

distance of the self pattern furthest from the ALC. Figure 5 shows that, with positive selection, the self pattern with the largest hamming distance to the ALC is used to determine the minimum  $a_{pos}$ .



**Figure 5:** Venn-diagram of Adapted Positive Selection ALC

Figures 4 and 5 also illustrate the drawback of *false positives* and *false negatives* when the respective selection methods train an ALC with an incomplete static self set. In Figures 4 and 5, the *known self* is the incomplete static self set that is used to train the ALC. The *unknown self* is the self patterns that are not known during training or represents self patterns which are outliers to the set of *known self* patterns.

### C. Matching a Non-self Pattern

The natural immune system has the ability to memorise frequently detected antigen. A B-Cell that frequently detects antigen goes into memory status. The B-Cells with memory status are used by the immune system in a secondary response to detect antigen much faster than the primary response. In order to model the NIS's memory ability, it is important to keep record of the number of non-self pattern matches (or detections) made by an ALC. After classifying a number of non-self patterns, the number of non-self matches

by an ALC will determine the ALC's matching ratio. The *Hit Counter (HC)* of an ALC is updated to keep record of the number of matches. The hamming distance,  $\gamma(\mathbf{x}, \mathbf{r})$ , between a pattern and the ALC's receptor can be used to determine if the ALC detects a non-self pattern, defined as follows

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

where  $\mathbf{x}$  is the bit-string of the pattern that needs to be classified,  $\mathbf{r}$  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. There are certain requirements for an ALC to be able to detect a non-self pattern. These requirements together with how the *HitCounter* of the ALC is updated, are explained in section IV(C.1) and section IV(C.2) for the adapted negative and positive selection methods respectively.

### 1) Adapted Negative Selection Hit Counter

An ALC which is trained with the adapted negative selection method can detect all non-self patterns which are closer to the ALC than the closest self pattern to that ALC (as discussed in section IV(B.1)). This means that the relation,  $\gamma(\mathbf{x}, \mathbf{r}) < a_{neg}$ , must hold for an ALC to be able to detect a non-self pattern. The value of  $\gamma(\mathbf{x}, \mathbf{r})$  indicates the specificity with which an ALC's receptor matched a non-self pattern.  $\gamma(\mathbf{x}, \mathbf{r}) \rightarrow 0$  indicates a more exact match to the pattern of the receptor and  $\gamma(\mathbf{x}, \mathbf{r}) \rightarrow a_{neg}$  a less exact match. 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.  $\gamma(\mathbf{x}, \mathbf{r}) \rightarrow 0$ ,  $1.0 - \frac{\gamma(\mathbf{x}, \mathbf{r})}{a_{neg}} \rightarrow 1.0$ ) than an ALC where  $\gamma(\mathbf{x}, \mathbf{r}) \rightarrow a_{neg}$ , 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* for adapted negative selection is calculated as

$$HC = HC + 1.0 + \left(1.0 - \frac{\gamma(\mathbf{x}, \mathbf{r})}{a_{neg}}\right), 0 < a_{neg} < k \quad (2)$$

with the initial value of  $HC = 0$ .

### 2) Adapted Positive Selection Hit Counter

For an ALC which is trained with the adapted positive selection method to detect a non-self pattern, the non-self pattern needs to be further away from the ALC than the furthest self pattern to that ALC (as discussed in section IV(B.2)). The following relation,  $\gamma(\mathbf{x}, \mathbf{r}) > a_{pos} \Rightarrow \gamma(\mathbf{x}, \mathbf{r}) - a_{pos} > 0.0$ ,

must hold for an ALC to detect a non-self pattern. As discussed in the previous section, the value of  $\gamma(\mathbf{x}, \mathbf{r})$  indicates the specificity with which an ALC's receptor matched a non-self pattern. 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* for adapted positive selection is calculated as

$$HC = HC + 1.0 + \left(\frac{\gamma(\mathbf{x}, \mathbf{r}) - a_{pos}}{k - a_{pos}}\right), 0 < a_{pos} < k \quad (3)$$

with the initial value of  $HC = 0$ .

### 3) The Hit Ratio

The *HitRatio (HR)* of an ALC is calculated after a specified number of patterns has been classified, referred to as the *IterationSize (IS)*. The *HitRatio (HR)* is calculated based on parameters *HC* and *IS*, as

$$HR(HC, IS) = \frac{HC}{IS} \quad (4)$$

*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.

### D. The Life Counter

Section IV(A) briefly introduced the life counter function and stated that the life counter (*LC*) determines in which state an ALC is at any given time. The *LC* maps the states to a continuous range, i.e.  $LC : \{low, medium, high\} \rightarrow (0, 1)$ . 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*. 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,  $\beta$ , of an ALC to detect a non-self pattern is calculated as follows:

- for an ALC trained with the adapted negative selection,

$$\beta_{neg} = 1.0 - \frac{a_{neg}}{k} \quad (5)$$

- for an ALC trained with the adapted positive selection,

$$\beta_{pos} = \frac{a_{pos}}{k} \quad (6)$$

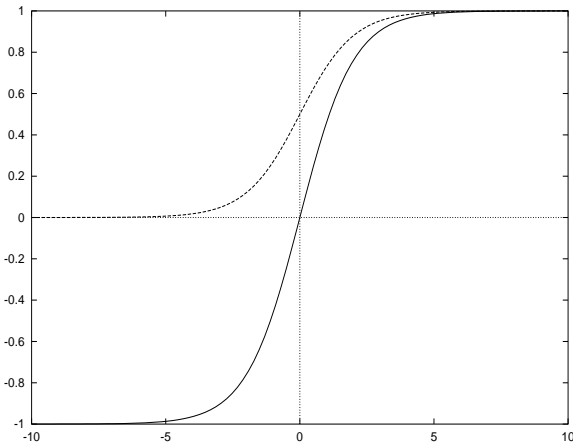
The ALC's life counter can be calculated after each  $IS$  patterns to determine the ALC's state. The next section explains the life counter threshold function that is used to determine the ALC's state.

### 1) Life Counter Threshold Function

After  $IS$  patterns have been classified, the LC threshold function,  $\tau$ , is calculated to determine the status of an ALC. The  $\tau$ -value of an ALC is based on the  $HR$ ,  $\beta$  and  $IS$  of the ALC. The requirement for  $\tau$  is that  $\tau \in (0, 1)$ , thus a function with a 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 illustrated by the dashed line in Figure 6, satisfies these requirements. The sigmoid function is defined as



**Figure 6:** Hyperbolic Tangent Function and Sigmoid Function

$$g(\lambda_{Sigmoid}, x_{Sigmoid}) = \frac{1}{1 + e^{(-\lambda_{Sigmoid} \times x_{Sigmoid})}} \quad (7)$$

where  $\lambda_{Sigmoid}$  controls the steepness of  $g(\lambda_{Sigmoid}, x_{Sigmoid})$ .

$g(\lambda_{Sigmoid}, x_{Sigmoid})$  is a monotonic increasing function if  $\lambda_{Sigmoid} > 0$  and a monotonic decreasing function if

$\lambda_{Sigmoid} < 0$ . Thus,

$$\tau(IS) = g(\lambda_{Sigmoid}, IS) \quad (8)$$

satisfies the requirements.

The steepness,  $\lambda_{Sigmoid}$ , represents the rate at which an ALC converges to memory or annihilated status. The  $HR$ -value of an ALC indicates the ratio at which an ALC matched non-self patterns (as explained in section IV(C.3)). Therefore, the  $HR$ -value determines the convergence rate at which an ALC converges to memory or annihilated status. An increasing  $HR$ -value implies an increasing  $\lambda_{Sigmoid}$ , which results in faster convergence of an ALC from mature to memory status. A decreasing  $HR$ -value implies a decreasing  $\lambda_{Sigmoid}$ , which results in faster convergence of an ALC from memory to mature or mature to annihilated status.

From the previous statements it can be derived that if  $HR > 0$ , then  $\lambda_{Sigmoid} > 0$  for  $\tau(IS)$  to be a monotonic increasing function, implying convergence to memory status. If  $HR < 0$ , then  $\lambda_{Sigmoid} < 0$  for  $\tau(IS)$  to be a monotonic decreasing function, implying convergence to annihilated status. If  $HR = 0$ , then  $\lambda_{Sigmoid} = 0$  for  $\tau(IS)$  to be a linear function, with  $\tau(IS) = 0.5$  for all  $IS$ , which implies that there is no convergence to memory or annihilation. To achieve one of the above situations, the steepness of the status transition function must be a function of  $HR$  with a range of  $(-1, 1)$ . The minimum matching ratio  $\beta$  of an ALC also influences the rate at which an ALC's status moves from one state to the next. The rate of a state transition should be faster for an increasing  $\beta$  than for a smaller value of  $\beta$ . A faster transition rate is obtained with a steeper gradient of the state transition function, which is achieved with  $\lambda_{Sigmoid} > 0$ . A slower transition is obtained for a smaller  $\beta$  by using a  $\lambda_{Sigmoid} < 0$ .

The requirements for  $\lambda_{Sigmoid}$  can be fulfilled if  $\lambda_{Sigmoid}$  is a function of the hyperbolic tangent function, defined as (see the solid line in Figure 6)

$$f(\lambda_{Hyper}, x_{Hyper}) = \frac{2}{1 + e^{(-\lambda_{Hyper} \times x_{Hyper})}} - 1 \quad (9)$$

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

$$\lambda_{Sigmoid} = f(\beta, HR) \quad (10)$$

$$= \frac{2}{1 + e^{(-\beta \times HR)}} - 1 \quad (11)$$

The steepness of the above hyperbolic tangent function is controlled by  $\beta$ . A large value of  $\beta$  causes a higher rate of change of  $f(\beta, HR)$ . This higher rate of change in  $f(\beta, HR)$  ensures an increasing slope for the state transition

function  $g$ . A smaller value of  $\beta$  causes a lower rate of change of  $f(\beta, HR)$ . The lower rate of change in  $f(\beta, HR)$  ensures a decreasing slope for the state transition function  $g$ . Values of  $HR > 0$  will cause a positive output for  $f(\beta, HR)$  to ensure a monotonic increasing state transition function. The state transition function will be a monotonic decreasing function for values of  $f(\beta, HR) < 0$ . The values of  $f(\beta, HR)$  will be negative if  $HR < 0$ .

In summary:

$$\tau(IS) \doteq g(\lambda_{Sigmoid}, IS) \quad (12)$$

$$= g(f(\beta, HR), IS) \quad (13)$$

$$= \frac{1}{1 + e^{(-f(\beta, HR) \times IS)}} \quad (14)$$

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

The definition of  $HR$  in section IV(C.3) where  $IS > 0$  and  $HC \geq 0$ , will result in  $HR$  to always have positive values. With  $HC \geq 0$  and  $\beta \geq 0$ ,  $f(\beta, HR) \geq 0$ . This results in a monotonic increasing function,  $g$ . This means that function  $g$  will always converge to 1.0 and all ALCs will either stay in the mature status or converge to memory status. To prevent all the ALCs to converge to memory, a constant detection ratio can be subtracted from  $HR$ . The constant detection ratio, referred to as the expected matching ratio ( $EMR$ ), is explained in the following section.

## 2) The Expected Matching Ratio

The expected matching ratio ( $EMR$ ) is a detection ratio to control the number of ALCs that would converge to memory (annihilated) status after classifying  $IS$  patterns, i.e. the detection ratio of non-self patterns expected from an ALC in an iteration of  $IS$  patterns. 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 in the next iteration of  $IS$  patterns. If  $\eta_h$  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. The  $EMR$  for the next iteration is updated to the average over the current matching ratio and the current iteration's  $EMR$ , i.e.

$$EMR_{h+1} = \frac{EMR_h + \frac{\eta_h}{IS}}{2.0} \quad (15)$$

where  $EMR_{h+1}$  is the calculated expected matching ratio for the next iteration. The  $HR$  function can now be redefined as

$$HR(HC, IS, EMR) = \frac{HC}{IS} - EMR \quad (16)$$

The redefined  $HR$  function implies 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.

## V. Genetic Algorithm to Evolve an ALC

Most AISs use a static set of randomly initialised ALCs, trained with either the negative selection method or positive selection method. The static set of ALCs is then used to detect non-self patterns. A drawback of using a static set of ALCs is the specified size of the set. If the size of the set is too large, there might be a higher average overlap among ALCs. The consequence of a small set size is the premature coverage of non-self space among ALCs. This section shows how a genetic algorithm (GA) can be used to evolve a dynamic set of ALCs. An advantage of a dynamic set of ALCs is that the size of the set is dynamically determined by the algorithm. The interested reader is referred to [2, 3, 24, 32] for more information on GAs. The evolved set of ALCs is used to detect non-self patterns. The success and efficiency of the GAIS is mainly influenced by the quality of the ALCs in the evolved set. Initially the active set of ALCs is empty. The purpose of the GA is to evolve one optimal ALC to be added to the active set of ALCs. Each evolved ALC considers the training patterns as well as the existing ALCs in the active set to satisfy the following objectives: the main objective of the GA is to evolve an ALC to be added to the existing active set of ALCs 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 adapted positive selection method. In addition to the main objective, the GA also needs to evolve an ALC to minimise the average overlap with the existing 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, forces greater coverage of non-self space, and minimises the number of ALCs in the active set.

In the case of the adapted negative selection method, 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 need to be maximised for maximum non-self space coverage. Thus, the GA used in GAIS, optimises multiple objectives [8].

The GA has an initial population of  $I$  randomly generated ALCs (as shown in Figure 7). The fitness of each ALC in the population is calculated to be proportional to the

ADT and overlap with other ALCs (refer to section V(C)). 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 operators and other design aspects of the algorithm are explained in more detail below.

The GAIS algorithm has two phases (as illustrated in Figure 7). In phase 1, the GA is repeatedly applied until the active ALC set in phase 1 has converged (as explained in section VI(B)). The GA in phase 1 of GAIS is summarised below:

1. set  $g=0$  ( $g$  is the generations counter)
2. randomly initialise  $I$  chromosomes as population  $H_g$
3. while  $H_g$  has not converged, or the maximum number of generations are not exceeded
  - (a) evaluate the fitness of each chromosome in  $H_g$  based on whether adapted negative or adapted positive selection is used
  - (b) apply cross-over as follows (as explained in section V(D))
    - i. select two parents
    - ii. apply uniform cross-over between the two selected parents and add the offspring to the set of offspring
  - (c) apply mutation as follows (as explained in section V(E))
    - 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 (as explained in section V(F))
  - (e)  $g = g + 1$

#### A. ALC as a Chromosome

Each chromosome in a population represents a potential solution, and consists of the set of parameters (genes) for which optimal values need to be found. *Alleles* are specific values from the domain of the corresponding parameter assigned to a gene. Thus a chromosome represents one ALC.

Each ALC has a bit-string receptor of fixed length,  $k$  (as explained in section IV). These receptors are used to detect or match patterns (as explained in section IV(C)). Therefore all patterns that need to be classified by an ALC's receptor must first be coded to a bit-string. Patterns with floating-point values are discretised using *binning*. *Binning* calculates the valid interval of values for each attribute  $c$

in the data set. The calculated interval of each attribute is divided into  $b$  bins, where each bin represents a group of values. The following equation is used to determine the *bin* of an arbitrary value,  $x_{c,j}$ ,

$$G(x_{c,j}) = \frac{x_{c,j} - \min_{c=1,\dots,C} \{x_{c,j}\}}{\max_{c=1,\dots,C} \{x_{c,j}\} - \min_{c=1,\dots,C} \{x_{c,j}\}} * b \quad (17)$$

where  $x_{c,j}$  is the floating-point value 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 which needs to be coded to binary. The number of groups or bins an attribute has, is used to determine the number of bits needed to represent the nominal values of an attribute. The number of bits per attribute is calculated as

$$N = \log_2 b \quad (18)$$

where  $b$  is the number of bins.

The binary encoded attribute values represent the receptor of an ALC.

#### B. The Initial Population

The initial population of the GA is a set of ALCs with randomly initialised receptors. 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.

#### C. Evaluating Fitness of Chromosomes

The ALCs in the AIS can be trained with one of two selection methods (as explained in section IV(B)). The fitness of an ALC is calculated based on the selection method used and by the average distance between the ALC and the existing set of ALCs, and the ALC's affinity distance threshold. The first objective to be optimised by the GA 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

$$\chi(D, \mathbf{r}) = \frac{\sum_{l=1}^{l \leq P} \gamma(\mathbf{d}_l, \mathbf{r})}{P} \quad (19)$$

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.  $\gamma$  is the hamming distance between  $\mathbf{d}_l$  and  $\mathbf{r}$ , defined as

$$\gamma(\mathbf{d}_l, \mathbf{r}) = \sum_{i=1}^{i \leq k} XOR(d_i, r_i) \quad (20)$$

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.

### 1) Adapted Negative Selection Fitness

As mentioned in the previous section an ALC can be trained with one of two selection methods. 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  $v_{neg}$  for an ALC trained with the adapted negative selection is defined as

$$v_{neg} = w_1 a_{neg} + w_2 \chi(D, \mathbf{r}) \quad (21)$$

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.

### 2) Adapted 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,  $v_{pos}$ , for an ALC trained with adapted positive selection is defined as

$$v_{pos} = w_1(k - a_{pos}) + w_2 \chi(D, \mathbf{r}) \quad (22)$$

where  $w_1$  and  $w_2$  have the same meaning as for adapted negative selection (explained in section V(C.1)).

### D. Parent Selection

Parent chromosomes are randomly selected from an elitist set of chromosomes [26]. These selected parent chromosomes produce a set of offspring through uniform crossover (as explained below). The elitist set of chromosomes consists of  $\varsigma$  best individuals in the population. The number of the individuals in the elitist set,  $\varsigma$ , is known as the *generation gap* and is calculated as

$$\varsigma = I * e \quad (23)$$

where  $e$  is the percentage of elite, with  $e \in (0, 1]$ .

The  $\varsigma$  selected individuals are the  $\varsigma$  best individuals that will survive to the next generation to ensure that the maximum fitness in the population does not decrease. 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 between randomly selected parents with probability,  $p_c$ , to produce offspring [19]. 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 the two parents, producing two 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.

$$p_c = 1.0 - \frac{g}{G} \quad (24)$$

where  $g$  is the completed number of generations and  $G$  is the maximum allowed number of generations. With an 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 more fit, the need to exchange genetic material between fit individuals decreases. Thus the probability of crossover decreases as the population evolves.

### E. Mutation

The created offspring (as explained above), is randomly selected for mutation. The GA applies random mutation on the selected individual with probability,  $p_m$  [19]. Random mutation selects genes randomly, and each gene's value is replaced with its complement (with probability  $p_m$ ). The probability,  $p_m$ , is calculated for each selected individual using

$$p_m = \frac{k - \chi(D, \mathbf{r})}{k} \quad (25)$$

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,  $p_m$  should be proportional to the similarity rate. 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.

### F. Selection of the New Population

The new population for the next generation is selected from both the parents of the current population and the offspring. As mentioned in the previous section, all individuals in the elitist set survive to the next generation without any mutation 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, where individuals are sorted according to their fitness [26]. Offspring with a high rank has a higher probability of being selected.

### G. Convergence of the GA

The GA is terminated 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  $\mu_T$ . The 2-point moving average is calculated as

$$\mu_g = \frac{\mu(H_{g-1}) + \mu(H_g)}{2.0} \quad (26)$$

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.  $\mu_g$  is calculated after each generation. The population  $H_g$  has converged if  $|\mu_{i-1} - \mu_i| < \mu_T$  for  $i = g - WindowSize, \dots, g$ . The *WindowSize* indicates the number of consecutive generations for which  $|\mu_{i-1} - \mu_i| < \mu_T$  must hold for the algorithm to converge. A small difference in moving average, i.e. less than  $\mu_T$ , 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.

## VI. The GAIS Algorithm

The GAIS algorithm has two phases (as illustrated in Figure 7). The GA in phase 1 of GAIS was explained in section V.

This section explains the second part of phase 1 and phase 2 of the classification process of GAIS.

The GAIS algorithm is illustrated in Figure 7 and 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) Use GA to evolve a new ALC
  - (c) Add best 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. Obtain an active set of ALCs,  $D$ , from phase 1
3. While not all patterns have been classified
  - (a) for  $i=0, \dots, IS - 1$ 
    - i. if a non-self pattern is not detected by active set  $D$ 
      - A. Try to detect the non-self pattern with annihilated set  $A$
      - B. Move annihilated ALCs that detected non-self patterns from  $A$  to  $D$
    - ii. Update  $HC$  of each ALC in  $D$  that detected the non-self pattern
  - (b) for  $l=0, \dots, |D| - 1$ 
    - 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  $D$  to  $A$ .
  - (d) Update the  $EMR$  (as explained in section IV(D.2))

### 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 step 2 in phase 1. These aspects are discussed in the following sections.

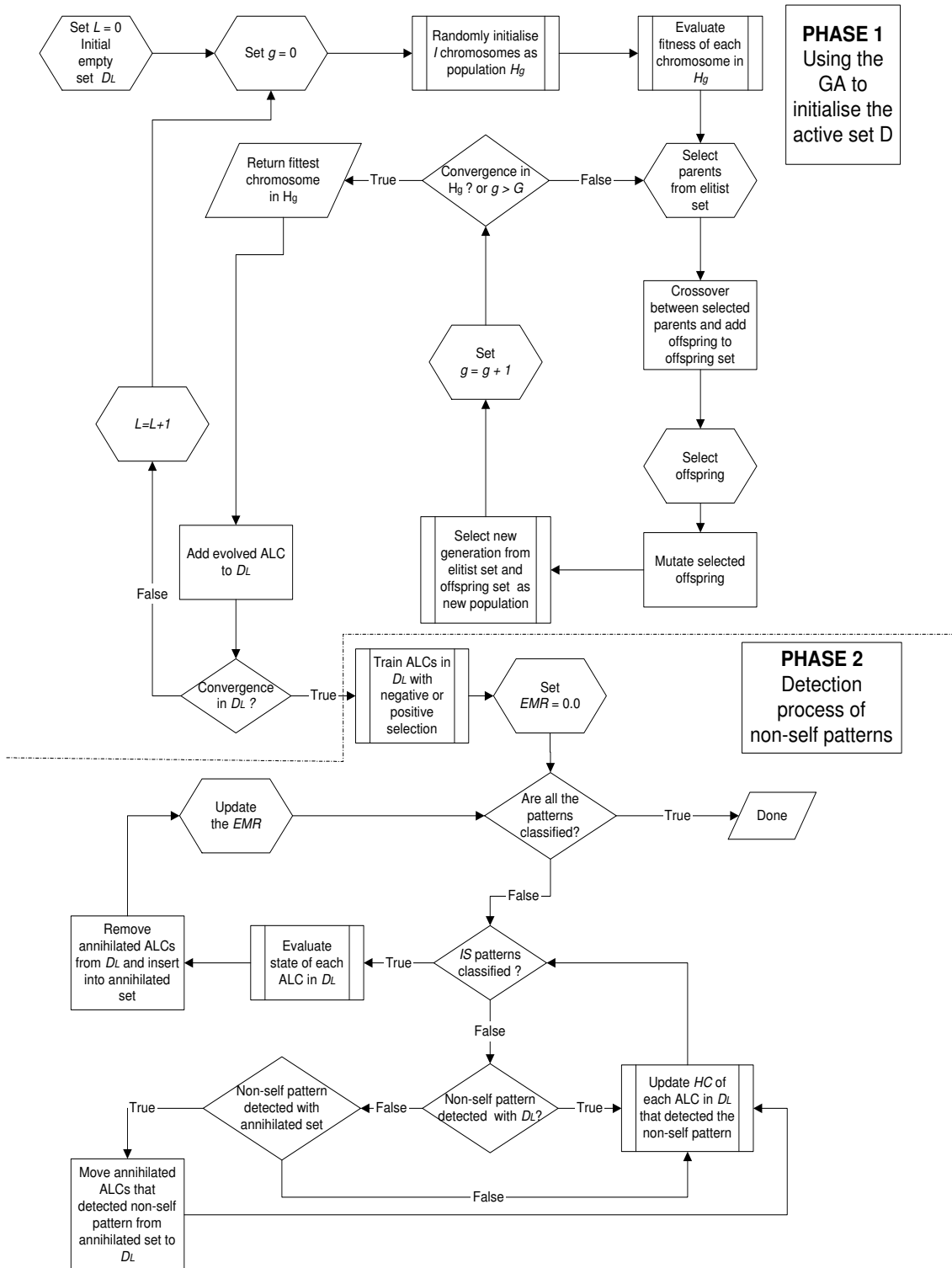


Figure. 7: Flow layout of the Artificial Immune System

### A. Initialisation of the ALC Set

The initial active ALC set is defined as the empty set  $D_0 = \{\}$ . The best ALC obtained from the GA is added to  $D_L$  until step 2 in phase 1 converges. The active set of ALCs,  $D_L$ , therefore grows i.e.

$$D_L = D_{L-1} \cup \{ALC_l\} \quad (27)$$

where  $ALC_l$  is the best ALC obtained from the GA and  $l = 0, \dots, L$ .  $L$  is the number of iterations before convergence in phase 1 is reached. The active set  $D_L$  is used by the fitness function in the GA to ensure that the newly evolved  $ALC_l$  is on average the furthest away from the active ALC set  $D_L$ .

### B. Convergence of Phase 1 in GAIS

In GAIS, 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 than  $\mu_T$ . The 2-point moving average is calculated as follows

$$\mu_L = \frac{\mu(D_{L-1}) + \mu(D_L)}{2.0} \quad (28)$$

where  $\mu(D_{L-1})$  is the average fitness of set  $D_{L-1}$  and  $\mu(D_L)$  is the average fitness of set  $D_L$ ;  $\mu_L$  is calculated after each evolved ALC is added to  $D_{L-1}$ . Convergence is reached when  $|\mu_{l-1} - \mu_l| < \mu_T$  for  $l = L - WindowSize, \dots, L$ . After convergence, all the ALCs in the active set  $D_L$  are trained with one of the selection methods (as explained in section IV(B)).

### C. Linking Phase 1 and Phase 2

The GA in phase 1 is repeatedly applied until convergence in phase 1, step 2 is reached (as explained in section VI(B)). The ALCs in the active set,  $D$ , are then trained in phase 2 with one of the selection methods (as explained in section IV(B)). The  $EMR$  is set to 0.0. The GAIS algorithm then uses the trained active set  $D$  to try and detect a set of patterns equal to the iteration size,  $IS$ , as non-self patterns. If a pattern was not detected by  $D$ , the annihilated set of ALCs is used to find a match. If an ALC in the annihilated set matches the pattern as non-self, the annihilated ALC becomes mature and is moved to  $D$ . The hit counter,  $HC$ , of each ALC in  $D$  is updated if the ALC detected a pattern as non-self (as explained in section IV(C)). After all the patterns in an iteration have been classified, the life counter,  $\tau(IS)$ , of each ALC in the active set  $D$  is calculated to determine in which state an ALC is. The ALCs that have annihilated status are removed from the active set  $D$  and inserted into the annihilated set which contains all previously annihilated ALCs. The  $EMR$  is then updated (as explained in section IV(D.2)). The next set of patterns equal to  $IS$  is then classified by GAIS until all the patterns have been classified.

## VII. Experimental Results

This section presents and discusses the experimental results 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 [5]. The patterns in each data set were discretised and converted to binary strings (as explained in section V(A)). For each experiment, one of the classes of 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 adapted 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. For each experiment the initial population size of the GA was set to 100 chromosomes ( $I = 100$ ) and the percentage of elite 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. All experimental results are averages over 30 simulations with the selected self class in *italic* print.

Tables 1 to 5 show the parameter settings for each dataset to obtain the best results. The results in tables 1 to 5, starting with the leftmost column, are the selected class as self, 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 separation 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. The average number of misclassified patterns for a specific parameter setting is calculated as

$$\#Misclassified = falsePositives + falseNegatives \quad (29)$$

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

$$IS = is * \text{Size of data set}, is \in [0.25, 0.50, 0.75, 100] \quad (30)$$

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

$$w_1 = \frac{W1}{100}, W1 \in [25, 50, 75, 100] \quad (31)$$

#### A. 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.

Table 1 shows the parameter settings for IS and W1 to obtain the best classification results for the different classes in the Iris data set. As a first experiment, *setosa* was selected as self. The best result for training the ALCs with negative selection on *setosa* as self, is with IS=37 and W1=50 where the average number of ALCs in the active set of ALCs (#ALCs = 16.427) and the false positive classification (fPos = 0.500) was the lowest. This gives an error 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. When training the ALCs with positive selection on *setosa* as self, a correct classification of 99.622% is obtained, with IS=37 and W1=50. The average number of ALCs with memory status in the active set of ALCs was 13.760. The results that follow in table 1 are for *versicolor* and *virginica* as selected self. These selected self classes have a correct classification of 97.133% and 94.600% with negative selection and 97.000% and 94.622% with positive selection.

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.

#### B. 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 therefore left out in the training and testing of the GAIS model. There are 16 missing attribute values for the bare nuclei attribute in the data set. The missing values were represented by binary strings as straight 1's. 458 patterns are of the benign class and 241 patterns of the malignant class. The patterns were converted to binary strings of length 36.

Table 2 shows the results for classifying the Wisconsin breast cancer data set with patterns of the *benign* class and *malignant* class as the self set. 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, with 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, with 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 obtained a correct classification of 93.395% with an average number of 33.033 ALCs, and an average HD of 13.041. The positive selection method obtained correct classification of 93.948% with an average number of 22.308 ALCs and 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 is distributed over a larger area than patterns from the *malignant* class, thus more ALCs are 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.

Table 1: Results for Iris data set

Class as self	IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
Negative Selection								
<i>Setosa</i>	37	50	16.427 ( $\pm 5.187$ )	12.980 ( $\pm 4.203$ )	0.500 ( $\pm 1.306$ )	0.000 ( $\pm 0.000$ )	10.874 ( $\pm 0.180$ )	8.012 ( $\pm 0.188$ )
<i>Versicolor</i>	37	75	21.567 ( $\pm 4.337$ )	17.220 ( $\pm 3.483$ )	0.967 ( $\pm 2.883$ )	3.333 ( $\pm 1.918$ )	10.943 ( $\pm 0.117$ )	6.805 ( $\pm 0.243$ )
<i>Virginica</i>	150	75	23.533 ( $\pm 5.198$ )	0.000 ( $\pm 0.000$ )	1.200 ( $\pm 3.210$ )	6.900 ( $\pm 3.942$ )	8.634 ( $\pm 0.162$ )	9.179 ( $\pm 0.197$ )
Positive Selection								
<i>Setosa</i>	37	50	17.420 ( $\pm 4.409$ )	13.760 ( $\pm 3.524$ )	0.567 ( $\pm 1.478$ )	0.000 ( $\pm 0.000$ )	9.043 ( $\pm 0.196$ )	7.948 ( $\pm 0.209$ )
<i>Versicolor</i>	37	75	20.333 ( $\pm 4.700$ )	16.253 ( $\pm 3.750$ )	1.167 ( $\pm 2.842$ )	3.333 ( $\pm 1.768$ )	9.041 ( $\pm 0.118$ )	6.757 ( $\pm 0.256$ )
<i>Virginica</i>	112	75	23.833 ( $\pm 4.793$ )	11.083 ( $\pm 2.271$ )	1.267 ( $\pm 3.194$ )	6.800 ( $\pm 4.552$ )	11.342 ( $\pm 0.127$ )	9.171 ( $\pm 0.172$ )

Table 2: Results for Wisconsin Breast Cancer data set

Class as self	IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
Negative Selection								
<i>Benign</i>	524	25	34.567 ( $\pm 5.029$ )	16.867 ( $\pm 2.526$ )	1.367 ( $\pm 1.189$ )	6.267 ( $\pm 5.219$ )	16.061 ( $\pm 0.188$ )	17.499 ( $\pm 0.053$ )
<i>Malignant</i>	699	100	33.033 ( $\pm 10.791$ )	0.000 ( $\pm 0.000$ )	1.767 ( $\pm 1.251$ )	44.400 ( $\pm 36.541$ )	16.267 ( $\pm 0.136$ )	13.041 ( $\pm 0.587$ )
Positive Selection								
<i>Benign</i>	524	25	33.800 ( $\pm 3.398$ )	16.467 ( $\pm 1.756$ )	1.500 ( $\pm 1.075$ )	6.900 ( $\pm 3.284$ )	19.922 ( $\pm 0.117$ )	17.490 ( $\pm 0.044$ )
<i>Malignant</i>	175	75	22.308 ( $\pm 5.555$ )	11.525 ( $\pm 3.397$ )	1.967 ( $\pm 1.377$ )	40.333 ( $\pm 19.752$ )	20.043 ( $\pm 0.121$ )	14.322 ( $\pm 0.267$ )

### C. Mushroom

The mushroom data set contains descriptions of artificially generated samples of 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. The missing values were represented by binary strings as straight 1's. 4208 patterns are of the edible class and 3916 patterns of the poisonous class. The patterns were converted to binary strings of length 57.

Table 3 summarises the results obtained with the mushroom data set. 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 for both selection methods is more influenced by their HD than by their ADT. The HD of both selection methods differs 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, 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 patterns from the *edible* class as self or patterns from the *poisonous* class as self.

### D. 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 pattern consists of 9 continuous valued attributes. The patterns were converted to binary strings of length 45. Table 4 shows the best classification results obtained for the Glass data set with patterns of the different classes as the self set. Both the classification results for ALCs trained with positive selection and negative selection are shown in table 4.

The results in table 4 show that in most cases, except in the case of the *building-window-float* class as self, the positive

Table 3: Results for Mushroom data set

Class as self	IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
Negative Selection								
<i>Edible</i>	4062	25	46.267 ( $\pm 3.999$ )	19.850 ( $\pm 2.297$ )	1.400 ( $\pm 2.222$ )	932.667 ( $\pm 196.295$ )	25.520 ( $\pm 0.131$ )	27.698 ( $\pm 0.063$ )
<i>Poisonous</i>	8124	25	45.033 ( $\pm 4.582$ )	0.000 ( $\pm 0.000$ )	1.700 ( $\pm 2.003$ )	1591.933 ( $\pm 382.550$ )	25.523 ( $\pm 0.142$ )	27.729 ( $\pm 0.049$ )
Positive Selection								
<i>Edible</i>	8124	25	43.933 ( $\pm 7.506$ )	0.000 ( $\pm 0.000$ )	1.200 ( $\pm 1.495$ )	1019.333 ( $\pm 373.054$ )	31.502 ( $\pm 0.153$ )	27.690 ( $\pm 0.059$ )
<i>Poisonous</i>	2031	100	50.717 ( $\pm 13.291$ )	17.242 ( $\pm 5.158$ )	2.467 ( $\pm 3.391$ )	1695.00 ( $\pm 401.959$ )	25.665 ( $\pm 0.178$ )	20.201 ( $\pm 0.336$ )

Table 4: Results for Glass data set

Class as self	IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
Negative Selection								
<i>Building-window-float</i>	106	25	40.444 ( $\pm 4.020$ )	25.233 ( $\pm 2.644$ )	1.367 ( $\pm 1.974$ )	14.167 ( $\pm 4.662$ )	21.946 ( $\pm 0.146$ )	21.735 ( $\pm 0.051$ )
<i>Building-window-nonfloat</i>	160	25	39.700 ( $\pm 3.064$ )	18.683 ( $\pm 1.774$ )	1.433 ( $\pm 1.794$ )	33.200 ( $\pm 7.889$ )	21.125 ( $\pm 0.134$ )	21.836 ( $\pm 0.048$ )
<i>Containers</i>	214	75	35.067 ( $\pm 10.392$ )	0.000 ( $\pm 0.000$ )	0.433 ( $\pm 2.373$ )	0.000 ( $\pm 0.000$ )	28.335 ( $\pm 3.152$ )	16.437 ( $\pm 1.157$ )
<i>Headlamps</i>	53	25	37.693 ( $\pm 4.189$ )	27.133 ( $\pm 6.224$ )	0.967 ( $\pm 5.295$ )	0.367 ( $\pm 0.615$ )	23.146 ( $\pm 4.130$ )	21.739 ( $\pm 0.160$ )
<i>Tableware</i>	53	50	30.220 ( $\pm 7.324$ )	23.280 ( $\pm 7.316$ )	0.300 ( $\pm 1.643$ )	0.000 ( $\pm 0.000$ )	28.786 ( $\pm 3.072$ )	18.268 ( $\pm 0.824$ )
<i>Vehicle-window-float</i>	106	25	39.989 ( $\pm 3.608$ )	25.289 ( $\pm 5.337$ )	0.567 ( $\pm 3.104$ )	2.267 ( $\pm 1.337$ )	23.941 ( $\pm 3.980$ )	21.737 ( $\pm 0.153$ )
Positive Selection								
<i>Building-window-float</i>	53	25	39.167 ( $\pm 3.570$ )	28.333 ( $\pm 2.746$ )	1.467 ( $\pm 1.978$ )	14.900 ( $\pm 4.722$ )	23.011 ( $\pm 0.208$ )	21.681 ( $\pm 0.099$ )
<i>Building-window-nonfloat</i>	160	25	39.800 ( $\pm 3.671$ )	18.700 ( $\pm 1.720$ )	1.367 ( $\pm 2.157$ )	31.900 ( $\pm 7.303$ )	23.879 ( $\pm 0.119$ )	21.846 ( $\pm 0.052$ )
<i>Containers</i>	53	25	38.013 ( $\pm 6.682$ )	29.113 ( $\pm 6.343$ )	0.433 ( $\pm 2.373$ )	0.000 ( $\pm 0.000$ )	21.125 ( $\pm 3.992$ )	21.152 ( $\pm 3.146$ )
<i>Headlamps</i>	214	25	39.000 ( $\pm 2.407$ )	0.000 ( $\pm 0.000$ )	0.967 ( $\pm 5.295$ )	0.467 ( $\pm 0.937$ )	21.959 ( $\pm 4.150$ )	21.803 ( $\pm 0.144$ )
<i>Tableware</i>	106	50	26.300 ( $\pm 8.422$ )	17.233 ( $\pm 6.229$ )	0.300 ( $\pm 1.643$ )	0.000 ( $\pm 0.000$ )	16.251 ( $\pm 3.077$ )	17.786 ( $\pm 1.966$ )
<i>Vehicle-window-float</i>	106	25	39.967 ( $\pm 5.840$ )	25.933 ( $\pm 5.424$ )	0.567 ( $\pm 3.104$ )	2.000 ( $\pm 1.287$ )	21.080 ( $\pm 3.983$ )	21.239 ( $\pm 2.596$ )

selection method obtained 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 obtained 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.

### E. Car Evaluation

The car evaluation data set was derived from a simple hierarchical decision model that was developed by [6]. 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.

The results in table 5 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 in comparison with the other classes as the selected 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 (therefore 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 obtained 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.

### F. Comparing the Results

The classification results obtained from the GAIS with the adapted negative and positive selection methods are summarised and compared with C4.5 in table 6. 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 obtained on average better classification than C4.5 in classifying the Iris data set, except for *virginica* as self. C4.5 obtained better classification for the Mushroom data set and the Car evaluation data set, where GAIS obtained better classification with the Glass data set. C4.5 obtained better classification with *malignant* as self and GAIS obtained 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 the adapted negative selection and the high average ADT for the adapted 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 than C4.5. This deduction supports the *no free lunch theorem* [59].

## VIII. Conclusion and Future work

This paper gave a brief overview on the classical view of the natural immune system (NIS) and also discussed the danger and network theory of NIS. Some of the existing AIS models and applications thereof was presented. The main objective of ALCs with maximum non-self coverage and least overlap among the ALCs, was addressed by presenting GAIS. GAIS uses a genetic algorithm to evolve artificial lymphocytes (ALCs) with least overlap among existing ALCs and maximum non-self coverage. Thus, the evolved ALC was a local optimum in the search space that was not covered by the existing set of ALCs. Each evolved ALC was added to the set of existing ALCs. The ALCs were trained with the adapted negative or the adapted 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. Annihilated ALCs were removed from the active set of ALCs. Therefore the life counter function

Table 5: Results for Car Evaluation data set

Class as self	IS	W1	#ALCs	#MemALCs	#fPos	#fNeg	ADT	HD
Negative Selection								
Acceptable	864	50	11.967 ( $\pm 1.520$ )	5.900 ( $\pm 0.803$ )	1.433 ( $\pm 2.208$ )	915.133 ( $\pm 49.171$ )	3.673 ( $\pm 0.048$ )	5.747 ( $\pm 0.037$ )
Good	432	50	17.642 ( $\pm 4.835$ )	12.675 ( $\pm 3.657$ )	0.433 ( $\pm 1.832$ )	230.233 ( $\pm 75.159$ )	5.320 ( $\pm 0.086$ )	4.836 ( $\pm 0.135$ )
Unacceptable	864	75	11.567 ( $\pm 2.622$ )	5.783 ( $\pm 1.311$ )	7.067 ( $\pm 6.746$ )	426.733 ( $\pm 33.595$ )	4.044 ( $\pm 0.133$ )	3.005 ( $\pm 0.241$ )
Very Good	432	50	17.992 ( $\pm 5.647$ )	13.200 ( $\pm 4.251$ )	0.333 ( $\pm 0.884$ )	129.133 ( $\pm 45.808$ )	5.719 ( $\pm 0.113$ )	4.532 ( $\pm 0.191$ )
Positive Selection								
Acceptable	864	25	15.533 ( $\pm 3.767$ )	7.567 ( $\pm 1.870$ )	1.367 ( $\pm 2.157$ )	897.733 ( $\pm 88.295$ )	8.542 ( $\pm 0.070$ )	5.950 ( $\pm 0.012$ )
Good	1296	50	16.833 ( $\pm 4.609$ )	8.383 ( $\pm 2.288$ )	0.367 ( $\pm 1.474$ )	243.000 ( $\pm 66.330$ )	6.701 ( $\pm 0.088$ )	4.874 ( $\pm 0.085$ )
Unacceptable	1296	75	12.400 ( $\pm 4.407$ )	6.200 ( $\pm 2.203$ )	6.467 ( $\pm 7.281$ )	425.233 ( $\pm 36.141$ )	7.956 ( $\pm 0.154$ )	3.034 ( $\pm 0.245$ )
Very Good	864	50	18.233 ( $\pm 5.399$ )	9.067 ( $\pm 2.648$ )	0.333 ( $\pm 0.922$ )	130.033 ( $\pm 61.279$ )	6.288 ( $\pm 0.106$ )	4.553 ( $\pm 0.181$ )

Table 6: Summarised results

Class as self	GAIS		C4.5
	Negative selection	Positive selection	
Iris - <i>Setosa</i>	99.66%	99.62%	99.3%
Iris - <i>Versicolor</i>	97.13%	97.0%	96.0%
Iris - <i>Virginica</i>	94.6%	94.62%	97.3%
Breast Cancer - <i>Benign</i>	98.907%	98.798%	96.3%
Breast Cancer - <i>Malignant</i>	93.395%	93.948%	96.3%
Mushroom - <i>Edible</i>	88.502%	87.438%	99.9%
Mushroom - <i>Poisonous</i>	80.383%	79.105%	99.9%
Glass - <i>Building-window-float</i>	92.741%	92.357%	79.3%
Glass - <i>Building-window-nonfloat</i>	83.816%	84.454%	79.7%
Glass - <i>Containers</i>	99.797%	99.797%	95.9%
Glass - <i>Headlamps</i>	99.376%	99.329%	94.9%
Glass - <i>Tableware</i>	99.85%	99.85%	97.7%
Glass - <i>Vehicle-window-float</i>	98.675%	98.8%	91.7%
Car - <i>Acceptable</i>	46.959%	47.968%	95.1%
Car - <i>Good</i>	86.651%	85.916%	98.7%
Car - <i>Unacceptable</i>	74.895%	75.017%	99.5%
Car - <i>Very Good</i>	92.507%	92.455%	100%

dynamically determined the number of ALCs in the active set. Results of GAIS on different data sets were presented and compared with C4.5. These results showed that there are cases that GAIS performs better than C4.5 and that the overlap among self and non-self patterns influences the number of evolved ALCs and the classification performance of GAIS. A drawback of GAIS is that the algorithm is restricted to two-class classification problems. GAIS is also not very scalable, since the complete dimension of a problem space is used to generate an ALC. Thus the time and computational complexity increases as the dimension of a classification problem increases. Future work will investigate the possibility to extend GAIS to multi-class classification and also investigate the possibility for a more scalable model. An in-depth analysis on the life counter function and the problem of overfitting the training data also needs to be investigated.

## References

- [1] T. Bäck, H. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.
- [2] D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms : Part I, Fundamentals*, University Computing, 1993.
- [3] D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms : Part II, Research Topics*, University Computing, 1993.
- [4] C.M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.
- [5] 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.
- [6] M. Bohanec, V. Rajkovic, "Expert System for Decision Making", *Sistemica*, vol. 1, no. 1, pp. 145-157, 1990.
- [7] 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.
- [8] C.A. Coello Coello, D.A. Van Veldhuizen, G.B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Plenum Pub Corp, 2002.
- [9] D. Dasgupta, *Artificial Immune Systems and their Applications*, ed., Berlin: Springer, 1998.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] L.N. de Castro, F.J. Von Zuben, "An Evolutionary Immune Network for Data Clustering", *In Proceedings of the IEEE Brazilian Symposium on Artificial Neural Networks (SBRN'00)*, pp. 84-89, 2000.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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 (ICANN'01)*, pp. 126-129, 2001.
- [18] 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.
- [19] A.P. Engelbrecht, *Computational Intelligence : An Introduction*, Wiley & Sons, 2002.
- [20] A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, Wiley & Sons, 2005.
- [21] S. Forrest, S. Hofmeyr, A. Somayaji, "Computer Immunology" (DRAFT), *Communications of the ACM*, vol. 40, no. 10, pp. 88-96, 1997.
- [22] 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.
- [23] 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.
- [24] A.S. Fraser, "Simulation of Genetic Systems by Automatic Digital Computers", *Australian Journal of Biological Science*, vol. 10, pp. 484-491, 1957.
- [25] 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.

- [26] G.E. Goldberg, K. Deb, "A Comparative Analysis of Selection Schemes used in Genetic Algorithms", In *G.J.E. Rawlins, ed., Foundations of Genetic Algorithms*, pp. 69-93, Morgan Kaufmann, 1991.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.
- [33] J.E. Hunt, D.E. Cooke, "Learning using an Artificial Immune System", *Journal of Network and Computer Applications*, vol. 19, pp. 189-212, 1996.
- [34] N.K. Jerne, "Towards a network theory of the immune system", *Annals of Immunology (Inst. Pasteur) 125C*, pp. 373-389, 1974.
- [35] 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.
- [36] J. Kennedy, R. C. Eberhart, "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, vol. 4, pp. 1942-1948, 1995.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, 1995.
- [41] P. Matzinger, "The Danger Model in its Historical Context", *Scandinavian Journal of Immunology*, vol. 54, pp. 4-9, 2001.
- [42] P. Matzinger, *The Real Function of the Immune System*, <http://cmmg.biosci.wayne.edu/asg/polly.html>, February 2004.
- [43] 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.
- [44] M. Oprea, A.S. Perelson, "Somatic Mutation leads to Efficient Affinity Maturation when Centrococytes recycle back to Centroblasts", *Journal of Immunology*, vol. 158, pp. 5155-5162, 1997.
- [45] M. Oprea, S. Forrest, "Simulated Evolution of Antibody Gene Libraries under Pathogen Selection", *1998 IEEE International Conference on Systems, Man and Cybernetics*, 1998.
- [46] A.S. Perelson, "Immune network theory", *Immunological Review*, vol. 110, pp. 5-36, 1989.
- [47] A.S. Perelson, G. Weisbuch, "Immunology for physicists", *Reviews of Modern Physics*, vol. 69, no. 4, October 1997.
- [48] 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.
- [49] 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.
- [50] L. Schindler, D. Kerrigan, J. Kelly, "Understanding the Immune System - Tutorial", *Science behind the news - National Cancer Institute*.
- [51] A. Somayaji, S. Hofmeyr, S. Forrest, "Principles of a Computer Immune System", *1997 New Security Paradigms Workshop*, pp. 75-82, 1998.
- [52] A. Somayaji, S. Forrest, *Automated Response Using System-Call Delays*, Usenix 2000.
- [53] J. Timmis, *Artificial immune systems: A novel data analysis technique inspired by the immune network theory*, Ph.D. thesis, University of Wales, Aberystwyth, 2001.

- [54] J. Timmis, M. Neal, "Investigating the Evolution and Stability of a Resource Limited Artificial Immune System", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 40-41, 2000.
- [55] 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.
- [56] 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.
- [57] 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.
- [58] 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.
- [59] D.H. Wolpert, W.G. Macready, "No Free Lunch Theorems for Search", Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1996.
- [60] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, vol. 8, pp. 338-353, 1965.