Artificial Immune Recognition System (AIRS): Revisions and Refinements

Andrew Watkins

Computing Laboratory
University of Kent at Canterbury, UK
and
Department of Computer Science
Mississippi State University, USA.
abw5@ukc.ac.uk

Jon Timmis

Computing Laboratory
University of Kent at Canterbury, UK
J.Timmis@ukc.ac.uk

Abstract

This paper revisits the Artificial Immmune Recognition System (AIRS) that has been developed as an immune-inspired supervised learning algorithm. Certain unnecessary complications of the original algorithm are discussed and means of overcomming these complexities are proposed. Experimental evidence is presented to support these revisions which do not sacrifice the accuracy of the original algorihtm but, rather, maintain accuracy whilst increasing the simplicity and data reduction capabilities of AIRS.

1 INTRODUCTION

Recently, there has been a great deal of interest in the use of the immune system as inspiration for computer science and engineering. These Artificial Immune Systems (AIS) seem to have great potential, which is as yet unrealized. An intuitive application of AIS is in the area of computer security, network intrusion detection (Forrest, Perelson et al. 1994), (Hofmeyr and Forrest 2000) and (Kim and Bentley 2001), change detection, and so on. However, AIS are not limited to this field alone. Work has identified that the immune system contains certain properties that may be useful to create learning algorithms for computer science through the exploitation of the natural learning mechanisms contained within the immune system (Bersini and Varela 1990). However, the focus of current AIS research seems to have been on the development of unsupervised learning algorithms (De Castro and Von Zuben 2000b) and (Timmis and Neal 2001) rather than the supervised or reinforcement kind. An exception to this is work in (Carter 2000). Recent work in (Watkins 2001) explored the possibility of utilizing the immune system as inspiration for the creation of a supervised learning technique. By extracting useful metaphors from the immune system and building on previous immune inspired unsupervised learning algorithms, a classifier was constructed that seems to perform reasonably well on various classification and machine learning problems (Watkins and Boggess 2002a).

This paper presents a further investigation into the work of (Watkins 2001) and suggests improvements to the algorithm that are capable of maintaining classification accuracy, whilst improving performance in terms of computational costs and an increase in the data reduction capabilities of the algorithm. This paper outlines the previous work undertaken in (Watkins 2001), suggests improvements to the algorithms and discusses the implications of these new results. Attention is then given to future possibilities with this approach.

2 BACKGROUND RESEARCH ON AIRS

AIRS (Artificial Immune Recognition System) is a novel immune inspired supervised learning algorithm (Watkins 2001). Motivation for this work came from the author's identification of the fact that there was a significant lack of research that explored the use of the immune system metaphor for supervised learning; indeed, the only work identified was that of (Carter 2000). However, it was noted that within the AIS community there had been a number of investigations on exploiting immune mechanisms for unsupervised learning (that is, where the class of data is unknown a-priori) (Timmis, Neal et al. 2000), (Timmis and Neal 2001) and (De Castro and Von Zuben 2000b). Work in (De Castro and Von Zuben 2000a) examined the role of the clonal selection process within the immune system (Burnet 1959) and went on to develop an unsupervised learning known as CLONALG. This work was extended by employing the metaphor of the immune network theory (Jerne 1974) and then applied to data clustering. This led to the development of the aiNet algorithm (De Castro and Von Zuben 2000b). Experimentation with the aiNet algorithm revealed that evolved *artificial immune networks*, when combined with traditional statistical analysis tools, were very effective at extracting interesting and useful clusters from data sets. aiNet was further extended to multimodal optimization tasks (De Castro and Timmis 2002b). Other work in (Timmis, Neal et al. 2000) also utilized the immune network theory metaphor for unsupervised learning, and then augmented the work with the development of a resource limited artificial immune network (Timmis and Neal 2001), which reported good benchmark results for cluster extraction and exploration with *artificial immune networks*. Indeed, this work has been further extended by (Nasaroui, Gonzalez et al. 2002) with the introduction of fuzzy logic and refinement of various calculations. The work in (Timmis and Neal 2001) was of particular relevance to (Watkins 2001) and the further work described in this paper.

Building on this previous work, in particular the ideas of artificial recognition balls and resource limitation from (Timmis and Neal 2001) and long-lived memory cells from (De Castro and Von Zuben 2000b). AIRS demonstrated itself to be an effective classifier. The rest of this section describes the immune metaphors that have been employed within AIRS, outlines the algorithm and discusses results obtained, before progressing to the following section, which describes augmentations and improvements to AIRS.

2.1 IMMUNE PRINCIPLES EMPLOYED

A little time should be taken to draw attention to the most relevant aspects of immunology that have been utilized as inspiration for this work. A more detailed overview of the immune system and its relationship with computer science and engineering can be found in (De Castro and Timmis 2002a).

Throughout a person's lifetime, the body is exposed to a huge variety of pathogenic (potentially harmful) material. The immune system contains lymphocyte cells known as B- and T-cells, each of which has a unique type of molecular receptor (location in a shape space). Receptors in this shape space allow for the binding of the pathogenic material (antigens), with the higher affinity (complementarity) between the receptor and antigen indicating a stronger bind. Work in (De Castro and Timmis 2002a) adopted the term shape-space to describe the shape of the data being used, and defined a number of affinity measures, such as Euclidean distance, which can be used to determine the interaction between elements in the AIS. Within AIRS (and most AIS techniques) the idea of antigen/antibody binding is employed and is known as *antigenic presentation*. When dealing with learning algorithms, this is used to implement the idea of matching between training data (antigens) and potential solutions (B-Cells). Work in (Timmis and Neal 2001) employed the idea of an *artificial recognition ball (ARB),* which was inspired by work in (Farmer, Packard et al. 1986) describing antigenic interaction within an immune network. Simply put, an ARB can be thought to represent a number of identical B-Cells and is a mechanism employed to reduce duplication and dictate survival within the population.

Once the affinity between a B-Cell and an antigen has been determined, the B-Cell involved transforms into a plasma cell and experiences *clonal expansion*. During the process of clonal expansion, the B-Cell undergoes rapid proliferation (cloning) in proportion to how well it matches the antigen. This response is antigen specific. These clones then go through *affinity maturation*, where some undertake somatic hypermutation (mutation here is inversely proportional to antigenic affinity) and eventually will go through a selection process through which a given cell may become a memory cell. These memory cells are retained to allow for a faster response to the same, or similar, antigen should the host become re-infected This faster response rate is known as the secondary immune response. Within AIRS, the idea of clonal expansion and affinity maturation are employed to encourage the generation of potential memory cells. These memory cells are later used for classification.

Drawing on work from (Timmis and Neal 2001), AIRS utilized the idea of a stimulation level for an ARB, which, again, was derived from the equations for an immune network described in (Farmer, Packard et al. 1986). Although AIRS was inspired by this work on immune networks, it was found that maintaining a network representation—with connections, stimulation, and repression among the ARBs in the system—was not necessary for evolving a useful classifier. In AIRS, ARBs experience a form of clonal expansion after being presented with training data (analogous to antigens); details on this process are provided in section 2.2. However, AIRS did not take into account the affinity proportional mutation. When new ARBs were created, they were subjected to a process of random mutation with a certain probability and were then incorporated into the memory set of cells should their affinity have met certain criteria. Within the AIRS system, ARBs competed for survival based on the idea of a resource limited system (Timmis and Neal 2001). A predefined number of resources existed, for which ARBs competed based on their stimulation level: the higher the stimulation value of an ARB the more resources it could claim. ARBs that could not successfully compete for resources were removed from the system. The term *metadynamics* of the immune system refers to the constant changing of the B-Cell population through cell proliferation and death. This was present in AIRS with the continual production and removal of ARBs from the population. Table 1 summarizes the mapping between the immune system and AIRS.

Table 1: Mapping between the Immune System and AIRS

IMMUNE SYSTEM	AIRS
Antibody	Feature vector
Recognition Ball	Combination of feature vector and vector class
Shape-Space	The possible values of the data vector
Clonal Expansion	Reproduction of ARBs that are well matched with antigens
Antigens	Training data
Affinity Maturation	Random mutation of ARB and removal of lowest stimulated ARBs
Immune Memory	Memory set of mutated ARBs
Metadynamics	Continual removal and creation of ARBs and Memory Cells

2.2 THE AIRS ALGORITHM

The previous section outlined the metaphors that were employed in the development of AIRS. This section now presents the actual algorithm and discusses the results obtained from experimentation. A more detailed description of the algorithm and results can be found in (Watkins 2001).

Within AIRS, each element (ARB) corresponds to a vector of *n* dimensions and a class to which the data belongs. Additionally, each ARB has an associated stimulation level as defined in equation 1, where *x* is feature vector of the ARB, s^x is the stimulation of an ARB *x*, *y* is the training antigen, and affinity, in the current implementation, is a function that calculates the Euclidean distance:

$$s^x = \begin{cases} 1\text{-affinity}(x,y), \text{if class of } x \equiv \text{class of } y \\ \text{affinity}(x,y), \text{ otherwise} \end{cases} \quad (1)$$

Notionally, AIRS has four stages to learning: initialization, memory cell identification, resource competition and finally refinement of established memory cells. AIRS is a one-shot learning algorithm; therefore, the process described below is run for each antigenic pattern, one at a time. Each of these processes will be outlined with the algorithm summarized below.

Initialization of the system includes data pre-processing (normalization) and seeding of the system with randomly chosen data vectors. Assuming a normalized input training data set (antigens), data from that set are randomly selected to form the initial ARB population **P** and memory cells **M**. Prior to this selection, an affinity threshold is calculated; this threshold for the current implementation is the average Euclidean distance between each item in the training data set. This is then used to control the quality of the memory cells maintained as classifier cells in the system.

AIRS maintains a population of memory cells **M** for each class of antigen, which, upon termination of the algorithm, should have identified suitable memory cells to provide a generalized representation for each class of antigenic pattern. The first stage of the algorithm is to determine the affinity of memory cells to each antigen of that class. Then the highest affinity cells are selected for cloning to produce a set of ARBs (which will ultimately be used to create an established memory set). The number of clones that are produced is in proportion to the antigenic affinity, i.e., how well they match; the ARBs also undergo a random mutation to introduce diversification.

The next stage is to identify the strongest, based on affinity to the training instance, ARBs; these will be used to create the established memory set used for classification. This is achieved via a resource allocation mechanism, taken from (Timmis and Neal 2001), where ARBs are allocated a number of resources based on their normalized stimulation levels. At this point, it is worth noting that the stimulation level of an ARB is calculated not only from the antigenic match, but also the class of the ARB. This, in effect, provides reinforcement for ARBs that are of the same class as the antigenic pattern being learnt and that match the antigenic pattern well, in addition to providing reinforcement for those that do not fall into that class and do not match the pattern well.

Once the stimulation of an ARB has been calculated, the ARB is allowed to produce clones (which undergo mutation). The termination condition is then tested to discover if the ARBs are stimulated enough for training to cease on this antigenic pattern. This is defined by taking the average stimulation for the ARBs of each class, and if each of these averages falls above a pre-defined threshold, training ceases for that pattern. This ARB production is repeated until the stopping criteria are met. Once the criteria have been met, then the candidate memory cell can be selected.

A candidate memory cell is selected from the set of ARBs based on its stimulation level and class, with the most stimulated ARB of the same class as the antigen being selected as the candidate. If this candidate cell has a higher stimulation than any memory cell for that class in the established memory set **M**, then it is added to **M**. Additionally, if the affinity of this candidate memory cell with the previous best memory cell is below the affinity threshold, then this established memory cell is removed from the population and replaced by the newly evolved memory cell, thus achieving population control.

This process is then repeated for all antigenic patterns. Once learning has completed, the set of established memory cells **M** can be used for classification. The algorithm is presented below, in terms of immune processes employed.

1. *Initialization*: Create a random base called the memory pool (**M**) and the ARB pool (**P**).
2. *Antigenic Presentation*: for each antigenic pattern do:

   a) *Clonal Expansion*:

   For each element of **M** determine their affinity to the antigenic pattern, which resides in the same class. Select highest affinity memory cell (*mc*) and clone *mc* in proportion to its antigenic affinity to add to the set of ARBs (**P**)

   b) *Affinity Maturation*:

   Mutate each ARB descendant of this highest affinity *mc*. Place each mutated ARB into **P**.

   c) *Metadynamics of ARBs*:

   Process each ARB through the resource allocation mechanism. This will result in some ARB death, and ultimately controls the population. Calculate the average stimulation for each ARB, and check for termination condition.

   d) *Clonal Expansion and Affinity Maturation*:

   Clone and mutate a randomly selected subset of the ARBs left in **P** based in proportion to their stimulation level.

   e) *Cycle*:

   While the average stimulation value of each ARB class group is less than a given stimulation threshold repeat from step 2.c.

   f) *Metadynamics of Memory Cells*:

   Select the highest affinity ARB of the same class as the antigen from the last antigenic interaction. If the affinity of this ARB with the antigenic pattern is better than that of the previously identified best memory cell *mc* then add the candidate (*mc-candidate*) to memory set **M**. Additionally, if the affinity of *mc* and *mc-candidate* is below the affinity threshold, then remove *mc* from **M**.

3. *Cycle*. Repeat step 2 until all antigenic patterns have been presented.

2.3 RESULTS AND DISCUSSION

AIRS was tested on a number of benchmark data sets in order to assess the classification performance. This section will briefly highlight those results and discuss potential improvements for the algorithm, more details can be found in (Watkins and Boggess 2002a).

Once a set of memory cells has been developed, the resultant cells can be used for classification. This is done through a k-nearest neighbor approach. Experiments were undertaken using a simple linearly separable data set, where classification accuracy of 98% was achieved using a k-value of 3. This seemed to bode well, and further experiments were undertaken using the Fisher Iris data set, Pima diabetes data, Ionosphere data and the Sonar data set, all obtained from the repository at the University of California at Irvine (Blake and Merz 1998). Table 2 shows the performance of AIRS on these data sets, a full comparison table of AIRS and other techniques can be found in (Watkins and Boggess 2002a).

Table 2: AIRS Classification Results on Benchmark Data

IRIS	**IONOSPHERE**	**DIABETES**	**SONAR**
96.7	94.9	74.1	84.0

These results were obtained from averaging multiple runs of AIRS, typically consisting of three, or more, runs and five-way, or greater, cross validation. More specifically, for the Iris data set a five-fold cross validation scheme was employed with each result representing an average of three runs across these five divisions. To remain comparable to other experiments reported in the literature, the division between training and test sets of the Ionosphere data set as detailed in (Blake and Merz 1998) was maintained. However, the results reported here still represent an average of three runs. For the Diabetes data set a ten-fold cross validation scheme was used, again with each of the 10 testing sets being disjoint from the others and results were averaged over three runs across these data sets. Finally, the Sonar data set utilized the thirteen-way cross validation suggested in the literature (Blake and Merz 1998) and was averaged over ten runs to allow for more direct comparisons with other experiments reported in the literature. During the experimentation, it was noted by the authors that varying system parameters such as number of seed cells varied performance on certain data sets, however, varying system resources (i.e., the numbers of resources an ARB could compete for) seemed to have little affect. A comparison was made between the performance of AIRS and other benchmark techniques, where AIRS seemed not to outperform specialist techniques, but on more general purpose algorithms, such as C4.5, it did outperform.

Even though initial results from AIRS did look promising, it can be said there are a number of potential areas for simplification and improvement. There is clearly a need to understand exactly why and how AIRS behaves the way it does. This can be achieved through a rigorous analysis of the algorithm, examining the behavior of the ARB pool and memory set over time. To date, the focus has been primarily on the classification performance of AIRS. Indeed, the final chapter of (Watkins 2001) suggests that an investigation into the resource allocation mechanism would be a useful area of investigation. The majority of AIS techniques use the metaphor of somatic hypermutation or affinity proportional mutation. To date, AIRS does not employ this metaphor but instead uses a naïve random generation of mutations.

The remaining sections of this paper undertake these investigations and present a modified version of AIRS, which is more efficient in terms of ARB production, employs affinity proportional mutation and assess what, if any, difference this has made to the overall algorithm.

3 A MORE EFFICIENT AIRS

Motivated by the observations in (Watkins 2001), current work has focused on refining AIRS. This section details the observations that have been made through a thorough investigation into AIRS and how issues raised through these observations have been overcome.

3.1 OBSERVATIONS

3.1.1 The ARB Pool

A very crude visualization[1] was used to gain a better understanding of the development of the ARB pool. In AIRS there are 2 independent pools of cells, the memory cell pool and the ARB pool. The initial formulation of AIRS uses the ARB pool to evolve a candidate memory cell of the same class as the training antigen, which can potentially enter the memory cell pool. During this evolution, ARBs of a different class than the training antigen were also maintained in the ARB pool. The stimulation of an ARB was based both on affinity to the antigen and class, where highly stimulated ARBs were those of the same class as the antigen and that were “close” to the antigen, or of a different class and "far" from the antigen. However, the visualization revealed that during the process of evolving a candidate memory cell, there seems no need to maintain or evolve ARBs that are a different class than the training antigen. The point of the interaction of the ARB pool with the antigenic material is really only in evolving a good potential memory cell, and this potential memory cell **must** be of the same class as the training antigen. When observing the visualization for a while, it is possible to notice that there is a process of convergence by ARBs of the same class to the training antigen. Naturally, based on the reward scheme, ARBs of a different class are moving further away from the training antigen. However, this process essentially must start over for the introduction of each new antigen, and, therefore, previously existing ARBs are fairly irrelevant. Since there are 2 separate cell pools, with the true memory of the system only being maintained in the Memory Cell pool, maintaining any type of memory in the ARB pool is unnecessary. This change to the algorithm rather than being about resource allocation schemes as initially suggested in (Watkins 2001) is really a simplification to the algorithm, which is seen as a positive step. This simplification affects both memory usage and computational simplification, although this will not be discussed in this paper.

3.1.2 Mutation of Cells

Motivated by observing the success of other AIS work, as well as by some of the tendencies discussed in (Watkins 2001) and (Watkins and Boggess 2002b), attention was paid to the way in which mutation occured within AIRS. In these two works, the authors notice that some of the evolved memory cells do not seem as high-quality of classifier cells as some of the others. Additionally, it was observed that there seemed to be some redundancy in the memory cells that were produced. In (De Castro and Von Zuben 2000a) and other AIS work, mutation within an antibody or B-Cell is based on its affinity, with higher affinity cells being mutated less than lower affinity cells. These other AIS works have used this method of somatic hypermutation to a good degree of success. It was thought that embedding some of this approach in AIRS might result in higher quality, less redundant, memory cells. This approach was therefore adopted within AIRS.

3.2 AIRS: WHAT IS NEW?

For the remainder of this section changes that have been made to the AIRS algorithm are described. There then follows empirical results from the new formulation and discuss the implications of these results.

3.2.1 Memory Cell Evolution

In the newly formulated version of AIRS, candidate memory cell evolution is based only on ARBs of the same class as the training antigen. This means that ARBs in the ARB pool are no longer permitted to mutate class. Therefore, the ARB pool will only consist of ARBs that are of the same class as the training antigen. At the end of each antigenic presentation cycle, the pool can be either be cleared out, or the ARBs can stay in the pool. If the pool is not cleared out then it will contain ARBs of all potential classes. The algorithm is only reinforcing the class of the antigenic pattern, and therefore, all ARBs that are in the pool at the end of the antigenic cycle that are not of the same class as the antigenic pattern will be removed through the metadynamic process, as they are no longer rewarded with any resources. This is in contrast to the original formulation of AIRS in which the

[1] See http://www.cs.ukc.ac.uk/people/rpg/abw5/ARB_hundred.html

allocation of resources, and thus cellular reinforcement, was based on a stimulation value that was calculated as in Equation 1 (section 2.2). In that original version both ARBs “near” the antigen and of the same class as the antigen were rewarded and ARBS “far” from the antigen and of a different class than the antigen were rewarded. Also, ARBs were allowed to mutate their class values (mutate in this case means switching classes). In the newly proposed version of AIRS, only ARBs of the same class are rewarded and mutation of the class value is no longer permitted.

Based on this new formulation, the only user parameter changes that might need to be made is that the stimulation threshold could potentially need to be raised. Recall, that the stimulation threshold was used as a stopping criterion for training the ARB pool on an antigen. In order to stop training on an antigen the average normalized stimulation level had to exceed the stimulation threshold for each class group of ARBs. That is, in a 2-class problem, for example, the average normalized stimulation level of all class 0 ARBs had to be above the stimulation threshold, and the average normalized stimulation level of all class 1 ARBs has to be above the stimulation threshold. It was possible, and frequently the case in fact, that the average normalized stimulation level for the ARBs of the same class as the training antigen reached the stimulation threshold before the average normalized stimulation level of ARBs in different classes from the antigen. What this did, in effect, was allow for the evolution of even higher stimulated ARBs of the same class while they were waiting for the other classes to reach the stimulation threshold. By taking out these extra cycles of evolution through no longer worrying with ARBs of different classes, it is possible that the ARBs will not have converged "as much" as in the previous formulation. This can be overcome by raising the stimulation threshold and thus requiring a greater level of convergence.

3.2.2 Somatic Hypermutation

To explore the role of mutation on the quality of the memory cells evolved, the mutation routine was modified so that the amount of mutation allowed by a given gene in a given cell is dictated by its stimulation value. Specifically, the higher the normalized stimulation value, the smaller the range of mutation allowed. Essentially, the range of mutation for a given gene = 1.0 - the normalized stimulation value of the cell. Mutation is then controlled over this range with the original gene value being placed at the center of the range. This, in a sense, allows for tight exploration of the space around high quality cells, but allows lower quality cells more freedom to explore widely. In this way, both local refinement and diversification through exploration are achieved.

3.3 THE AIRS V2 ALGORITHM

The changes made to the AIRS algorithm are small, but end up having an interesting impact on both the simplicity of implementation and on the quality of results. Section 4 will offer more discussion by way of comparison. For now, the changes to the original AIRS presented in section 2.2 will be discussed. These can be identified as follows:

1. Only the Memory Cell pool is seeded during initialization rather than both the MC pool (**M**) and the ARB pool (**P**). Since we are no longer concerned about maintaining memory or class diversity within **P** it is no longer necessary to initialize **P** from the training data or from examples of multiple classes.
2. During the clonal expansion from the matching memory cell used to populate **P**, the newly created ARBs are no longer allowed to mutate class. Again, maintaining class diversity in **P** is not necessary.
3. Resources are only allocated to ARBs of the same class as the antigen and are allocated in proportion to the inverse of an ARB’s affinity to the antigen.
4. During affinity maturation (mutation), a cell’s stimulation level is taken into account. Each individual gene is only allowed to change over a finite range. This range is centered with the gene’s pre-mutation value and has a width the size of the difference of 1.0 and the cell’s stimulation value. In this way the mutated offspring of highly stimulated cells (those whose stimulation value is closer to 1.0) are only allowed to explore a very tight neighborhood around the original cell, while less stimulated cells are allowed a wider range of exploration. (It should be noted that during initialization all gene values are normalized so that the Euclidean distance between any two cells is always within one. During this normalization, the values to transform a given gene to within the range of 0 and 1 are discovered, as well. This allows for this new mutation routine to take place in a normalized space where each gene is in the range of 0 and 1.)
5. The training stopping criterion no longer takes into account the stimulation value of ARBs in all classes, but now only accounts for the stimulation value of the ARBs of the same class as the antigen. In the new formulation of AIRS it is still possible to have ARBs in **P** of different classes if the implementation does not clear the ARB pool after each antigenic pattern. However, this will not affect the stopping criterion since the changes to the algorithm now only require that the average stimulation value of the ARBs of the **same** class as the antigen be above the user-supplied stimulation threshold.

3.4 RESULTS AND DISCUSSION

To allow for comparison between the two versions of the algorithm, the same experiments were performed on the new formulation of AIRS (AIRS2). Section 4 will provide a more thorough comparative discussion, but for now, results of AIRS2 on the four, previously discussed, benchmark sets are presented in Table 3.

Table 3: AIRS2 Classification Results on Benchmark Data

IRIS	IONOSPHERE	DIABETES	SONAR
96.0	95.6	74.2	84.9

These results were obtained by following the same methodology as the original results reported in section 2.3 which is elaborated upon in (Watkins 2001) and (Watkins and Boggess 2002a). Again, we note that these results are competitive with other classification techniques discussed in the literature, such as C4.5, CART, and Multi-Layer Perceptrons.

4 COMPARATIVE ANALYSIS

This section briefly touches on some comparisons between the original version of AIRS presented in discussed in section 2 (AIRS1) and the revisions to this algorithm presented in section 3 (AIRS2). The focus of this discussion will be on two of the more important features of the AIRS algorithms: classification accuracy and data reduction.

4.1 CLASSIFICATION ACCURACY

The success of AIRS1 as a classifier (cf, (Watkins and Boggess 2002a)) makes it important to assess any potential changes to the algorithm in light of test set classification accuracy. To aid in this task, Table 4 presents the best average test set accuracies, along with the standard deviations, achieved by both versions of AIRS on the four benchmark data sets.

Table 4: Comparative Average Test Set Accuracies

	AIRS1: Accuracy	AIRS2: Accuracy
Iris	96.7 (3.1)	96.0 (1.9)
Ionosphere	94.9 (0.8)	95.6 (1.7)
Diabetes	74.1 (4.4)	74.2 (4.4)
Sonar	84.0 (9.6)	84.9 (9.1)

It can be noted that the revisions to AIRS presented in section 3 do not require a sacrifice in classification performance of the system. In fact, for 3 of the 4 data sets we see a slight improvement in the accuracy; however, these differences are not statistically significant. What is important to note is that the changes introduce no fundamental differences in classification accuracy of the system.

4.2 DATA REDUCTION

From the previous subsection it can be seen that the changes introduced to AIRS offer no real difference in classification accuracy, so the question arises: why bother? Why introduce these changes to a perfectly reasonably performing classification algorithm? The answer lies in the data reduction capabilities of AIRS.

In (Watkins 2001) and (Watkins and Boggess 2002b), the authors discuss that aside from competitive accuracies another intriguing feature of the AIRS classification system is its ability to reduce the number of data points needed to characterize a given class of data from the original training data to the evolved set of memory cells. Given the volumes of data involved with many real-world data sets of interest, any technique that can reduce this volume while retaining the salient features of the data set is useful. Additionally, it is this collection of memory cells that are the primary classifying agents in the evolved system. Since classification is, currently, performed in a *k*-nearest neighbor approach, whose classification time is dependent upon the number of data points used for classifying a previously unseen data item, any reduction in the overall number of evolved memory cells is also useful for the algorithm.

Table 5 presents the average size of the evolved set of memory cells and the amount of data reduction this represents in terms of population size and percentage reduction, along with standard deviations, for each version of the algorithm on the four benchmark data sets. The original training set size is also presented for comparison. There are two points of interest:

1. Both versions of the algorithm exhibit data reduction, and
2. AIRS2 tends to exhibit greater data reduction than AIRS1.

Table 5: Comparison of the Average Size of the Evolved Memory Cell Pool

	Training Set Size	AIRS1: Memory Cells	AIRS2: Memory Cells
Iris	120	42.1/65% (3.0)	30.9/74% (4.1)
Ionosphere	200	140.7/30% (8.1)	96.3/52% (5.5)
Diabetes	691	470.4/32% (9.1)	273.4/60% (20.0)
Sonar	192	144.6/25% (3.7)	177.7/7% (4.5)

		(3.7)	(4.5)

This second point is the more important for our current discussion. As mentioned in sections 3.1.2 and 3.2.2, one of the goals of the revision of the AIRS algorithm was to see if employing somatic hypermutation through a method more in keeping with other research in the AIS field would increase the efficiency of the algorithm. The current measure of efficiency under concern is the amount of data needed to represent the original training set to achieve accurate classifications. We can see from Table 5 that, in general, AIRS2 was able to achieve the comparable accuracy presented in section 4.1 with greater efficiency. In fact for some of the data sets, most notably Ionosphere and Diabetes, the degree of data reduction is greatly increased (from 30% to 52% for Ionosphere data and from 32% to 60% for the diabetes data set). Interestingly, for the most difficult classification task, the Sonar data set, the degree of data reduction is not increased. While this was not the general trend on this data set (data not presented), it does possibly point to some limitations in the current version of AIRS. Overall, however, it seems reasonable to claim that the revisions to AIRS provide greater data reduction, and hence greater efficiency, without sacrificing accuracy.

4.3 A WORD ABOUT SIMPLICITY

While the focus has not been on algorithmic complexity analysis of the two versions of AIRS for this current paper, it would be remiss not to make a brief mention concerning the simplifying effects of the revision to AIRS. As mentioned in section 3.1, the reformulation of AIRS was chiefly motivated by some basic observations about the workings of the system. One observation was that the original version of AIRS maintained representation of too many cells for its required task. This led to the elimination of maintaining multiple classes of cells in the ARB pool or of retaining cells in the ARB pool at all. This has the simplifying effect of reducing the memory necessary to run the system successfully. A second observation concerning the quality of the evolved memory cells led to the investigation of the mutation mechanisms employed in the original algorithm. By adopting an approach to mutation proven to be successful in other AIS, it has been possible to increase the quality of the evolved memory cells that is evidenced by the increased data reduction without a decrease in classification accuracy. Both of these overarching changes (ARB pool representation and the mutation mechanisms used) have exhibited a simplifying effect on the classification system as a whole.

5 CONCLUSIONS AND FUTURE WORK

This paper has focused on a supervised learning system based on immunological principles. The Artificial Immune Recognition System (AIRS) introduced in (Watkins 2001) exhibited initial success as a classification algorithm. However, as with any initial system, there were some revisions and refinements that could be made to AIRS that would decrease the complexity of the system. This paper has presented investigations for two of these revisions.

It was shown that the internal data representation of the original version of AIRS was overcomplicated. By simplifying the evolutionary process, it was possible to decrease this complexity whilst still maintaining accuracy. It was also shown that the use of affinity aware mechanisms of somatic hypermutation, as adopted throughout the AIS community, led to higher quality memory cells in AIRS and thus greater data reduction and faster classification of test data items.

Both of these revisions were the result of careful observation of the behavior of the original algorithm. In this respect, it can be said that this paper is also about the importance of taking the steps to investigate the behavior of a system even if it is performing in a successful manner. This paper has demonstrated that such an investigation is fruitful in simplifying the workings without sacrificing the performance of the system.

There are many avenues that can be explored with this work. One is the analogy of this work with reinforcement learning strategies, it could possibly be argued that AIRS is a reinforcement learning algorithm, when one considers certain mechanism within the immune system (Bersini and Varela 1994); this warrants further investigation. Additionally, the role of parallel and distributed processing could be examined, in order to allow for dealing with larger scale problems. Work has already begun on applying AIRS to immunological data, attempting to predict the binding of receptors and in effect trying to solve an immunological problem with an artificial immune system.

References

Bersini, H. and F. J. Varela (1990). Hints for Adaptive Problem Solving Gleaned from Immune Networks. *Parallel Problem Solving from Nature*. pp.343-355

Bersini, H. and F. J. Varela (1994). The Immune Learning Mechanisms: Reinforcement, Recruitment and their Applications. *Computing with Biological Metaphors*. R. Paton, Chapman and Hall**:** 166-192.

Blake, C. L. and C. J. Merz (1998). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

Burnet, F. M. (1959). The Clonal Selection Theory of Immunity, Vanderbilt University Press, Nashville, TN.

Carter, J. H. (2000). "The Immune System as a model for Pattern Recognition and Classification." *Journal of the American medical Informatics Association* **7**(1).

De Castro, L. N. and J. Timmis (2002b). "An Artificial Immune Network for Multimodal Optimisation." *Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence:* 699-704.

De Castro, L. N. and J. I. Timmis (2002a). Artificial Immune Systems: *A New Computational Intelligence Approach*, Springer-Verlag.

De Castro, L. N. and F. Von Zuben (2000a). "The clonal selection algorithm with engineering applications." *Proceedings of Genetic and Evolutionary Computation* Conference: 36-37.

De Castro, L. N. and F. Von Zuben (2000b). "An Evolutionary Immune Network for Data Clustering." SBRN '00 **1**: 84-89.

Farmer, J. D., N. H. Packard, et al. (1986). "The Immune System, Adaptation, and Machine Learning." *Physica* **22**(D): 187-204.

Forrest, S., A. Perelson, et al. (1994). "Self-Nonself Discrimination in a Computer." *Symposium on Research in Security and Privacy*: 202-212.

Hofmeyr, S. and S. Forrest (2000). "Architecture for an Artificial Immune System." *Evolutionary Computation* **7**(1): 45-68.

Jerne, N. K. (1974). "Towards a Network Theory of the Immune System." *Annals of Immunology* **125C**: 373-389.

Kim, J. and P. Bentley (2001). "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with Negative Selection Operator." *Congress on Evolutionary Computation*: 1244-1252.

Nasaroui, O., F. Gonzalez, et al. (2002). "The Fuzzy Artificial Immune System: Motivations, Basic Concepts and Application to Clustering and Web Profiling." *International Joint Conference on Fuzzy Systems*: 711-717.

Timmis, J. and M. Neal (2001). "A resource limited artificial immune system for data analysis." *Knowledge Based Systems* **14**(3-4): 121-130.

Timmis, J., M. Neal, et al. (2000). "An Artificial Immune System for Data Analysis." *BioSystems* **55**(1/3): 143-150.

Watkins, A. (2001). AIRS: A resource limited artificial immune classifier. Department of Computer Science, Mississippi State University. http://nt.library.msstate.edu/etd/show.asp?etd=etd-11052001-102048

Watkins, A. and L. Boggess (2002a). "A new classifier based on resource limited artificial immune systems." *Proceedings of Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence.*: 1546-1551.

Watkins, A. and L. Boggess (2002b). "A resource limited artificial immune classifier." *Proceedings of Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence*: 926-931.