CLONAL SELECTION THEORY & CLONALG

THE CLONAL SELECTION CLASSIFICATION ALGORITHM (CSCA)

Technical Report
No. 2-02

Jason Brownlee
Master of Information Technology, Swinburne University of Technology, 2004
Bachelor of Applied Science, Computing, Swinburne University of Technology, 2002
(jbrownlee@ict.swin.edu.au)

Centre for Intelligent Systems and Complex Processes (CISCP)
Faculty of Information & Communication Technologies (ICT)
Swinburne University of Technology (SUT)

January 2005

Abstract

The vertebrate immune system is a robust and powerful information process system that demonstrates features such as distributed control, parallel processing and adaptation and learning via experience. Artificial Immune Systems (AIS) are machine-learning algorithms that embody some of the principles and attempt to take advantages of the benefits of natural immune systems for use in tackling complex problem domains. The Clonal Selection Algorithm (CLONALG) is one such system inspired by the clonal selection theory of acquired immunity, which has shown success on broad range of engineering problem domains. The focus of this work is the CLONALG algorithm, specifically the techniques history, previous research and algorithm function. In addition, this work seeks to take desirable elements from CLONALG implementations and devise a clonal selection based classification algorithm. Competence with the CLONALG algorithm is demonstrated in terms of theory and application. The algorithm is analysed from the perspective of desirable features for an immune inspired classifier and a number a new clonal selection inspired technique called the Clonal Selection Classification Algorithm (CSCA) is proposed, designed, specified and preliminary tested. The technique is shown to be robust, self-tuning (in terms of resources used), insensitive to user parameters and competitive as a classification system on common problem domains. An implementation of the CSCA is provided as a plug-in for the WEKA machine-learning workbench, as is a naïve implementation of the CLONALG algorithm for classification.

Acknowledgements

Firstly, I would that to sincerely thank my advisor Professor Tim Hendtlass and the CISCP for giving me the opportunity and financial support to research and work in a field that is both interesting and practical, and something I am passionate about. Secondly, thanks to the support provided by all the people at CISCP, for listening to my rants and for consistently providing positive and constructive feedback. I would like to thank Simon Garrett for his assistance in providing me with electronic copies of difficult to obtain papers and articles.

Table of Contents

List of Figures

List of Equations

List of Tables

1. Introduction

Artificial immune systems are a technique new to the scene of biological inspired computation and artificial intelligence, based on metaphor and abstraction from theoretical and empirical knowledge of the vertebrate immune system. A robust biological process critical to the combating of disease in the body, the immune system is known to be distributed in terms of control, parallel in terms of operation, and adaptive in terms of function, all of which are features desirable for solving complex or intractable problems faced in the field of artificial intelligence. This document describes and reviews one implementation of artificial immune systems called CLONALG (CLONal selection ALGorithm) inspired by the clonal selection theory of acquired immunity. Given the background, theory and an application of CLONALG to engineering applications, a new clonal selection inspired classification algorithm called Clonal Selection Classification Algorithm (CSCA) is designed, specified in detail and preliminary tested.

Section 2. provides a descriptive overview of the primary elements of clonal selection theory, focusing specifically on those elements abstracted and implemented in the CLONALG technique. Section 3. provides a review of the CLONALG algorithm focusing on the techniques implementation details, basic function and user-defined parameters. Section 4. provides a complete (to the authors knowledge) history of CLONALG. The section provides additional details of the clonal selection theory abstracted in the technique, and indicates the reasons why specific design decisions were made with the techniques inception. A number of augmentations to the CLONALG algorithm are described, focusing on those features that that may prove interesting or useful for the development of a clonal selection based classification algorithm.

Finally, section 5. describes how the clonal selection theory can be used as the inspiration for a classification algorithm. The section starts with a review of an abstracted form of CLONALG and speculates at the source of the techniques power. The reviewed variants of CLONALG are then revisited, specifically the features exhibited of each of the techniques that would provide desirable in a classification system. Using CLONALG as a base, a new clonal selection inspired algorithm is designed called the Clonal Selection Classification Algorithm (CSCA). The implementation details of the technique are described and preliminary test results indicate a simple, robust and effective classifier system.

2. Clonal Selection Theory

The primary task of the immune system can be summarised as distinguishing between itself and non-self [1]. The immune system is an organ used to protect an organism from foreign material that is potentially harmful to the organism (pathogens), such as bacteria, viruses, pollen grains, incompatible blood cells and man made molecules. The immune system consists of two main parts, innate immunity which consists of basic elements the organism is born with such as skin and natural barriers, and acquired immunity located only in vertebrates (organisms with a spinal column/backbone). Acquired or learned immunity supplements innate immunity and is acquired through contact with antigens

(something that evokes an immune response). The following lists the primary features of acquired immunity taken from [1]:

Specificity – The systems able to discriminate between different molecular entities and specialise

Adaptiveness – The systems ability to respond to previously unseen molecules, even those molecules that previously never existed on earth (man made)

Discrimination between self and non-self – This critical element of the immune system indicates that the system is capable of recognising and responding to foreign elements, though is able to tell the difference between what is foreign and potentially harmful, and what is actually apart of its own system. Without this discrimination, the immune system would respond against the organism for which it exists to protect.

Memory – The systems ability to recall previous contact with foreign molecules and respond in a learned manner. This means that a larger and more rapid response occurs on subsequent encounters with an antigen, this effect is called an anamnestic response.

The clonal selection theory is a theory postulated by Burnet, Jerne, Talmadge, used to describe the functioning of acquired immunity, specifically a theory to describe the diversity of antibodies used defend the organism from invasion [2]. An antibody is a molecule produced by B lymphocyte cells that can neutralise a single antigen. Each B lymphocyte (white blood cell) creates unique or customised antibodies of a specific type. The theory, when originally proposed was a point of contention and competed with another model called template theory. Today, the clonal selection theory is seen as fact given the overwhelming amount of empirical evidence.

The theory specifies that the organism have a pre-existing pool of heterogeneous (individually unique) antibodies that can recognise all antigens with some level of specificity. When an antigen is matched (selects) an antibody, it causes the cell to chemically bind to the antigen, replicate and produce more cells with the same receptor. During the cell proliferation stage, genetic mutations occur in the clone of cells that promote the match or affinity with the antigen. This allows the binding ability of the cells to improve with time and exposure to the antigen. This selection of replication cells by antigens can be viewed as a type of Darwinian microcosm where the fittest cells (best match with antigens) are selected for survival, and genetic mutation provides cell variation.

The image below provides a good basic overview of the clonal selection process. The image shows B lymphocyte cells at the top that bind to specific antigens. Once bound, the cell proliferates (divides or mitosis) and produces many B lymphoblasts which differentiate into either plasma cells that produce antibodies (effector of the immune response), or long lived memory cells (used if the antigen reappears).

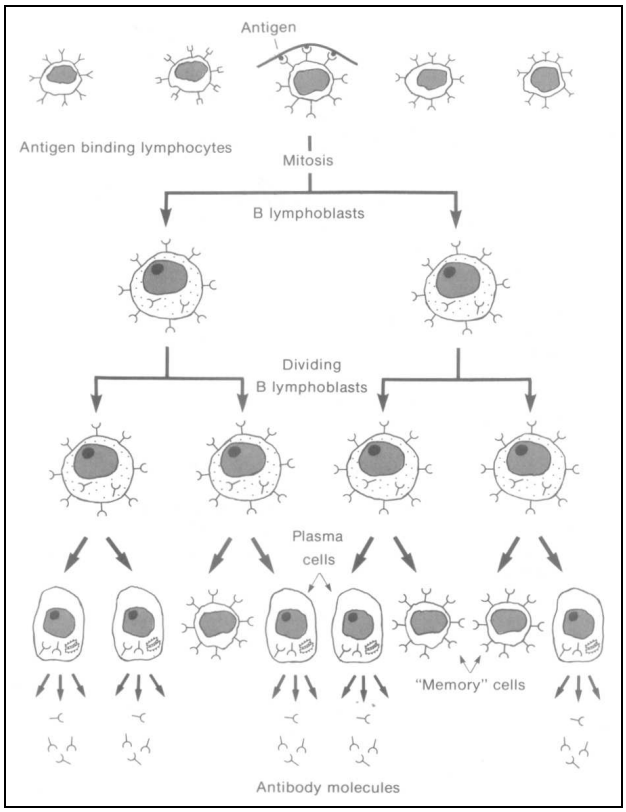


Figure 1 - Provides a simple overview of the clonal selection process, image taken from [3]

A further interesting aspect of acquired immunity is how the system develops the capability to differentiate between self and non-self. This phenomenon is called tolerance and describes the systems failure to launch an immunological response against a given antigen such as self-antigens. This ability is developed before birth of the organism, as the immune system itself develops. This ability to not generate antibodies against own cells, can be acquired for foreign antigens that are administered to the organism before birth. An aspect of this immunological tolerance is that it cannot be maintained if the antigen is not persistent in the organism, meaning the system can forget unused knowledge.

The figure below provides an additional schematic overview of the clonal selection process and how antigen tolerance is achieved. Here the figure shows the development of B lymphocyte cells and the negative selection against those cells that respond to self or are self-reactive.

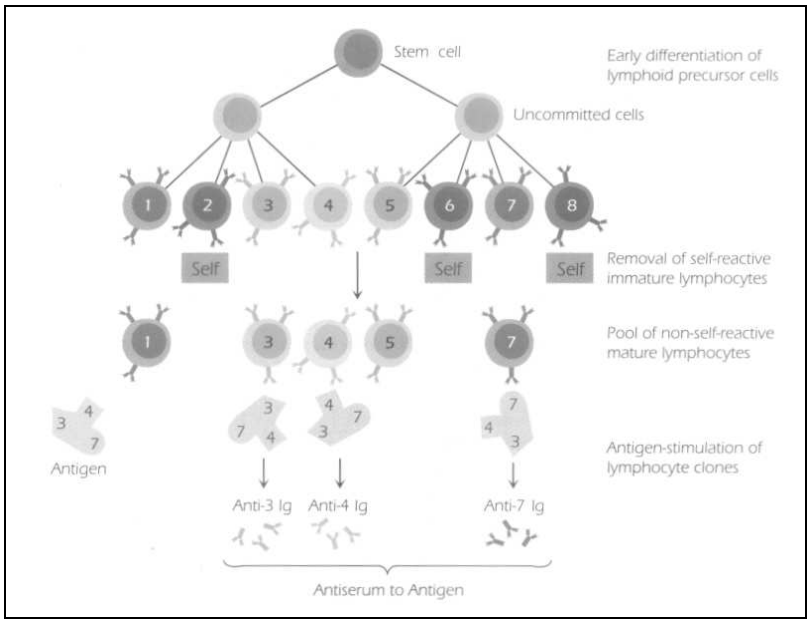


Figure 2 - Shows an overview of the clonal selection process and how antigen tolerance is achieved, taken from [1].

A final interesting aspect of the clonal selection theory of acquired immunity is that the system exhibits homeostasis. This means that when the system detects and responds to a foreign antigen, it is able to then return to a level of equilibrium. This implies that the system has an innate ability to regulate itself and the immunological responses provided.

It is recognised that an area of acquired immunity not discussed here is that of T lymphocyte cells and their role in both stimulating B lymphocyte cells, and their role in neutralising those cells infected with foreign antigens.

3. Clonal Selection Algorithm (CLONALG)

An artificial immune system technique that is inspired by the functioning of the clonal selection theory of acquired immunity is CLONALG (CLONal selection ALGorithm). This section provides an overview of CLONALG with a focus on the discrete functional units of the techniques implementation. Also provided in this section is an overview of the user parameters of the system and their effect on the functioning of the system.

3.1 Overview

The Clonal Selection Algorithm, originally called CSA in [4], and renamed to CLONALG in [5] is said to be inspired by the following elements of the clonal selection theory:

- Maintenance of a specific memory set
- Selection and cloning of most stimulated antibodies
- Death of non-stimulated antibodies
- Affinity maturation (mutation)

- Re-selection of clones proportional to affinity with antigen
- Generation and maintenance of diversity

The goal of the algorithm is to develop a memory pool of antibodies that represents a solution to an engineering problem. In this case, an antibody represents an element of a solution or a single solution to the problem, and an antigen represents an element or evaluation of the problem space.

The algorithm provides two mechanisms for searching for the desired final pool of memory antibodies. The first is a local search provided via affinity maturation (hypermutation) of cloned antibodies. More clones are produced for better matched (selected) antibodies, though the scope of the local search is inversely proportional to the selected antibodies rank. This allows those antibodies with low specificity with the antigen, a wider area in the domain which to maturate. The second search mechanism provides a global scope and involves the insertion of randomly generated antibodies to be inserted into the population to further increase the diversity and provide a means for potentially escaping local optima. The biologically links of CLONALG with the clonal selection theory of acquired immunity will be further discussed in section 4.

3.2 Algorithm

The following provides an overview of the steps of the CLONALG algorithm.

1. **Initialisation** – The first step of the CLONALG technique is initialisation, which involves preparing an antibody pool of fixed size ***N***. This pool is then partitioned into two components, a memory antibody section *m* that eventually becomes representative of the algorithms solution and a remaining antibody pool *r* used for introducing additional diversity into the system.
2. **Loop** – The algorithm then proceeds by executing a number of iterations of exposing the system to all known antigens. A single round of exposure or iteration is referred to as a generation. The number of generations ***G*** the system executes is user configurable, though the system can use a problem specific stopping condition.
    a. **Select Antigen** – A single antigen is selected at random without replacement (for the current generation) from the pool of antigens
    b. **Exposure** – The system is exposed to the selected antigen. Affinity values are calculated for all antibodies against the antigen. Affinity is a measure of similarity, and is problem dependent. It is common to use Hamming distance.
    c. **Selection** – A set of ***n*** antibodies are selected from the entire antibody pool that have the highest affinity with the antigen.
    d. **Cloning** – The set of selected antibodies are then cloned in proportion to their affinity (rank based).
    e. **Affinity Maturation (mutation)** – The clone (set of duplicate antigens) are then subjected to an affinity maturation process to better match the antigen in question. Here, the degree of maturation is inversely

proportional to their parent's affinity (rank based), meaning that the greater the affinity, the lower the mutation.

f. **Clone Exposure** – The clone is then exposed to the antigen, and affinity measures are calculated.
g. **Candidature** – The antibody or antibodies with the highest affinity in the clone are then selected as candidate memory antibodies for placement into *m*. If the affinity of a candidate memory cell is higher than that of the highest stimulated antigen from the memory pool *m*, then it replaces said antigen. Group replacements occur in a similar, but batched manner.
h. **Replacement** – Finally, the ***d*** individuals in the remaining *r* antigen pool with the lowest affinity are replaced with new random antibodies.

3. **Finish** – After the completion of the training regime, the memory *m* component of the antigen pool is then taken as the algorithms solution. Depending on the problem domain, the solution may be a single best individual antigen or the collective of all antigens in the pool.

The figure below provides a complementing diagrammatic representation of the tasks and workflow of the CLONALG technique.

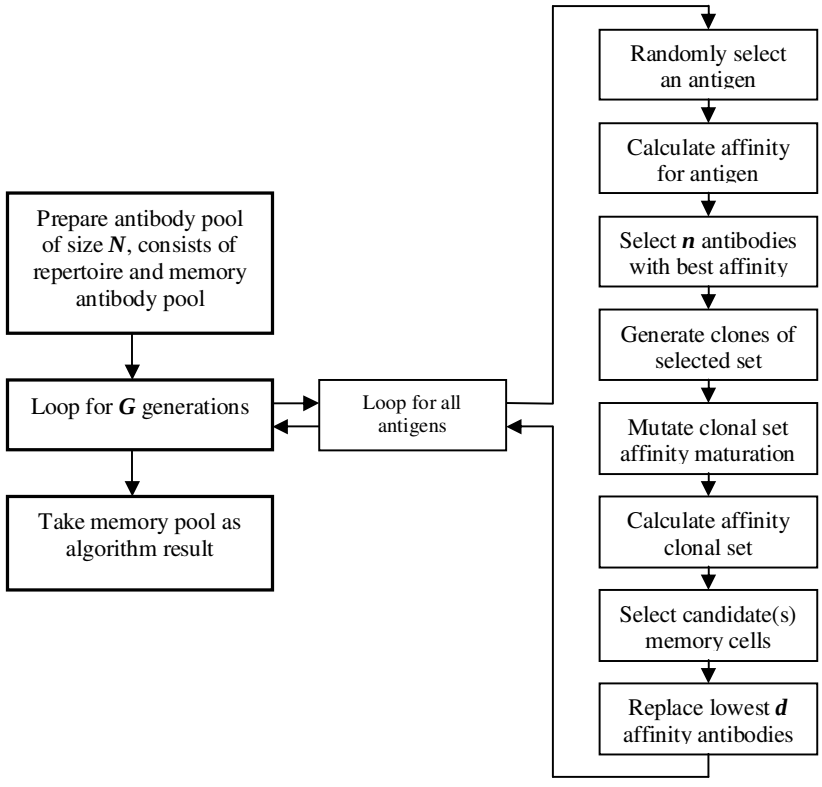


Figure 3 - Overview of the CLONALG algorithm

As mentioned, the number of clones created from each of the *n* selected antibodies is proportional to their affinity using a rank based measure. This is achieved by first sorting

the set of selected antibodies in ascending order by their affinity to the antigen. The ordered list is then iterated, and the number of clones created from each antibody is calculated as follows:

$$numClones = \left\lfloor \frac{\beta \cdot N}{i} + 0.5 \right\rfloor$$

Equation 1 - The number of clones created for each antibody

where β is a clonal factor, N is the size of the antibody pool, and i is the antibody current rank where $i \in [1, n]$. The total number of clones prepared for each antigen exposure to the system is thus calculated as:

$$Nc = \sum_{i=1}^{n} \left\lfloor \frac{\beta \cdot N}{i} + 0.5 \right\rfloor$$

Equation 2 - The total number of clones created for all antibodies per antigen exposure

where Nc is the total number of clones, and n is the number of selected antibodies. The specific means of performing affinity proportionate mutation is not specified for the general case, instead, like the representation, it is problem dependent. As mentioned, affinity is a similarity measure typically taken as the Hamming distance between two bit strings of equal length. Hamming distance is a simple measure that counts the number of point difference between two strings and is calculated as follows:

$$D = \sum_{i=1}^{L} \delta \text{ where } \delta = \begin{cases} 1 & ab_i \neq ag_i \\ 0 & ab_i \equiv ag_i \end{cases}$$

Equation 3 - Hamming distance calculation

where D is the Hamming distance, ab is the antibody, ag is the antigen and L is the length of the bit string. For a precise overview of the CLONALG algorithm with notation and pseud code, see [5].

3.3 Configuration

The CLONALG technique is relatively low in complexity and has a small number of user parameters when compared to other AIS systems such as AIRS (Artificial Immune Recognition System) [6]. This section provides an overview of the algorithms user configurable elements and their effect on the system.

3.3.1 Parameters

CLONALG has seven user defined parameters which include; antibody population size, memory pool size, selection pool size, remainder replacement size, clonal factor, number of generations and the random number generator seed (required for all techniques with a stochastic element).

Antibody population size (*N*) – Specifies the total amount of resources, that is the total number of antibodies to be maintained by the system. This includes both the memory antibody pool and the remainder antibody pool. The manner in which the antibody pool is divided is not specified. One simple approach is to simply select an m that defines the portion of N allocated to the memory pool, where $m \leq N$. This allows the size of the remainder to be defined as $r = N - m$.

Selection pool size (*n*) – Specifies the total number of antibodies with greatest affinity to draw from the antibody population for cloning on the presentation of each antigen, where $n \in [1, N]$. The value of n defines the amount of selective pressure on the population to reach high affinity values and thus proliferate. Smaller values increase the pressure and can decrease the diversity of the population by ensuring only the best affinity antibodies are cloned and replace elements of the memory pool.

Remainder replacement size (*d*) – Specifies the total number of lowest affinity antibodies to replace with random antibodies on each antigen exposure, where $d \in [0, r]$, and r is the size of remainder antibody pool. Provides a mechanism for introducing additional diversity (more than the local diversity of mutation), into the antibody population. This additional diversity can be disabled by setting the value to zero.

Clonal factor (β) – Specifies a scaling factor for the number of clones created for selected antibodies. Assuming an N value of 100, and a β value of 0.5, then using Equation 1, the number of clones created for the for the most stimulated antibody would be 200. Common values are $\beta \in (0,1]$. The lower the value, the more search in the local area (in relation to current antibodies) is performed by the algorithm.

Number of generations (*G*) – Specifies the total number of algorithm iterations to perform, where a single iteration sees the system exposed to all known antigens. The parameter controls the amount of learning the system will perform on the problem domain. Generation values to high may result in the system over-learning the problem or in being stuck on a locally optimal solution.

3.3.2 Sensitivity Analysis

For more information as to the effect of some of the parameters of the system, see [5] for a sensitivity analysis on multimodal function optimisation problem domains. The three parameters investigated were n (number of best affinity selected), Nc (total clones produced) and d (number of remainder replacements). Some of the interesting results revealed from the sensitivity analysis were as follows:

Sensitivity of *n* – It was shown that n does not strongly affect the number of generations until convergence. Increases in n, increased the computational complexity of the algorithm, thus slowing its execution. The average population affinity increased with n meaning there are more antibodies around the optimum in the memory pool.

Sensitivity of *Nc* – The total number of clones is controlled both by the number of antibodies selected for cloning (n) and the clonal scale factor (β). As the total number of clones increased, the time until converge (in terms of the number of generations decreased), though obviously the space complexity to store all the clones increased as does the time complexity to generate, maturate and evaluate the clones.

Sensitivity of *d* – The number of remainder antibody replaced with new random antibodies each antigen exposure had a strong effect on the diversity of the antibody population. Higher values result in a close to random search; therefore, values between 5% and 20% were suggested for common usage.

4. Evolution of CLONALG

This section provides a review of both the development of the CLONALG technique, and research into extending the technique both in application to additional problem domains and in making the technique more useable and efficient.

4.1 Metaphor and Abstractions

As has been mentioned the CLONALG technique is inspired by the clonal selection theory of acquired immunity. An overview of this theory has already been provided in section 2. The intent of this section is to provide some additional detail regarding this theory and more specifically, relate the theory to the design decisions made in the inception of CLONALG.

Affinity maturation is the foundation of both clonal selection theory and the CLONALG technique. Affinity maturation or cell maturation means the maturing of the system or maturing of the systems response to antigens over time. Further, affinity maturation implies learning by the system that requires at least two elements, namely variation and greediness. When a B lymphocyte cell chemically binds to an antigen to begin the proliferation and diversification process, the clone of cells produced must be similar to the primary cell, though must also incorporate variation in some way to ensure an overall improvement in response. In addition, the system is greedy in that there needs to be mechanisms to ensure only the higher affinity cells proliferate, and that some how the lower affinity cells do not.

Cell diversification is discussed in [4,5], where it is indicated that variation is achieved by two main processes that are then used as the basis for functions used in CLONALG, as follows:

Somatic hypermutation – This involves random genetic changes to the genes of the cloned cells that control its unique antigen receptor. The effect of this proliferation and variation is that high affinity variants are permitted in to the pool and the systems repertoire is increased through maturation. Cell mutation can be considered a form of local search, though needs some form of regulation. This abstraction was used as the basis for mutation in CLONALG, where the regulation mechanism used was the affinity between the cell and the antigen. Clearly what is needed for practical application of this abstraction, as indicated in [4] is a rapid response and cell maturation process

Receptor editing & new cells – It is indicated in [4], that recent immunological research shows that perhaps not all low affinity cells or cells that select self-antigens are deleted. It was said that some cells may be reused by the system, having their antigen receptors rewritten or edited. Also indicated is the fact the system does not rely solely on cell proliferation to manage the pool, and that nearly 5%-8% of the pool is periodically replaced with entirely new cells. These two abstractions were taken as the basis for the deletions and random antibody insertions that occur on each antigen exposure in CLONALG.

The technique purposefully does not attempt to model the cell-to-cell interactions of acquired immunity and does not differentiate a B lymphocyte cell and the antibodies it produces. Instead, the core principle of the system is that of a recognition unit responding in someway to an external antigen stimulus. For practical considerations, the technique uses a fixed sized pool of recognition cells, and given the greedy nature of the metaphor requires both that higher affinity cells dominate the response and are preserved, and that lower affinity cells are removed from the population. This need is the basis for the antibody selection process and the candidate replacement of low affinity antibodies in CLONALG.

The system can be considered associative. This means that those cells that are specialised to a specific antigen are also capable of responding to other antigens with similar characteristics. In immunology, this effect is referred to as cross-reaction or a cross-reaction response and is the property taken advantage of when an organism is inoculated with a vaccine. In CLONALG, this property can be referred to as generalisation where a single antibody can be an exemplar or a condensed form of information that the system has been exposed to.

The final point of mention is that the vertebrate immune system is typically exposed to the same antigen more than once. The adaptive capability of the system permits the system to reuse information, rather than start from scratch on each antigen presentation. The more exposure the system receives the better it learns. CLONALG can be considered to have a chronic infection where the number of generations determines the number of times the system is exposed to a specific antigen, providing a control over the amount of learning or adaptation the system undergoes.

Work by White and Garrett [7] provided a concise reiteration of the main areas of clonal selection exploited in artificial clonal selection algorithms, as follows:

Diversity – maintain a sparsely separated population distribution over a wide are of the antigenic space.

Optimisation – selection and maturation of high affinity antibodies results in a population that consist primary of high-quality solutions

Exploration – Mutation and other operations provide a means for exploration of the affinity landscape between antibodies and antigens

Replacement – ensures proliferation of higher affinity antibodies and the removal of lower affinity elements allowing a broader search of the affinity landscape

Reinforcement Learning – repeated exposure to an antigenic stimulus and selection reward those antibodies that respond the highest allowing the system to become more specific with time

4.2 Initial Applications

The inception of the Clonal Selection Algorithm (CSA) [4], and its subsequent revisit and renaming to CLONALG [5] showed an implementation and application of the technique to three forms of common engineering problem domains, namely pattern recognition, function optimisation and combinatorial optimisation. This section discusses these inaugural applications of CLONALG and the results observed.

4.2.1 Pattern Recognition

The pattern recognition problem addressed with CLONALG was that of binary character recognition. This involved a set of eight characters represented as a string of 120 binary values. The problem was designed to test the learning and memory acquisition aspects of the system. Over a number of generations, the system was able to provide increasingly accurate matches, eventually converging to a point were elements of the memory pool exactly represented the antigenic patterns the system was exposed to.

Figure 4 - Binary characters used for the pattern recognition problem, taken from [4]

The implementation of the algorithm involved selecting a reduced set of antibodies for cloning and maturation, then selecting a single best maturated antibody for placement in the memory population. The configuration used for the problem involved using a pool size of ten, eight of which made up the memory pool. No antibodies were replaced with random individuals (d=0), and five antibodies were selected for maturation for each antigen (n=5). Finally the clonal factor (β) used was 10. The system converged to the input patterns in 250 generations.

4.2.2 Function Optimisation

A two-dimensional multimodal function optimisation was used to test CLONALG in [4], and two further one-dimensional problems were tested in [5]. In these cases, affinity was taken as the function evaluation of the antibody in question, and the antibodies were taken as binary encoded parameters of the function. It was shown that CLONALG is an appropriate technique for function optimisation, specifically multimodal optimisation were the problem space consists of multiple desirable local optimum solutions.

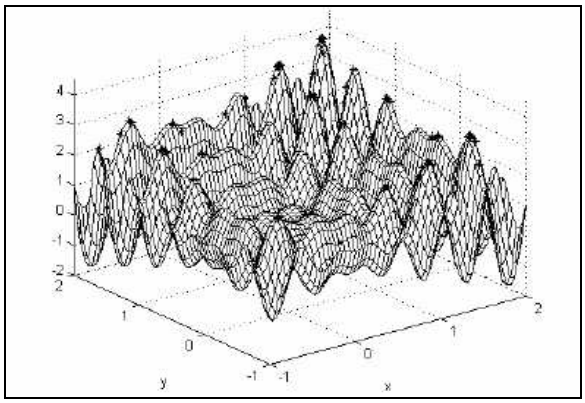


Figure 5 - Shows the antibody population on the tested 2D function, taken from [5]

In the application of optimisation problems, the goal of CLONALG is to find the extrema of some function (minimise or maximise), rather than recognise something. In this case, the entire pool of antibodies are selected for cloning and maturation, and a set of the best clones rather than the single best clone are selected for candidature in the memory set. For all tested functions the number of new cell replacements was zero (d=0), and the clonal factor (β) was 0.1. For the two-dimensional function, the antigen pool size was 100, whereas for the two one-dimensional functions the pool size was 50.

4.2.3 Combinatorial Optimisation

The combinatorial optimisation problem tested was a 30-city version of the widely known Travelling Salesperson Problem (TSP). The problem finding a tour where each city is visited once, and the last city connects to the first city visited, where each link in the tour has a cost. The goal is to simply minimise the cost of the tour. An integer representation of the problem was used where each antibody represented a single permutation of cities. The system converged on the best solution in 300 generations.

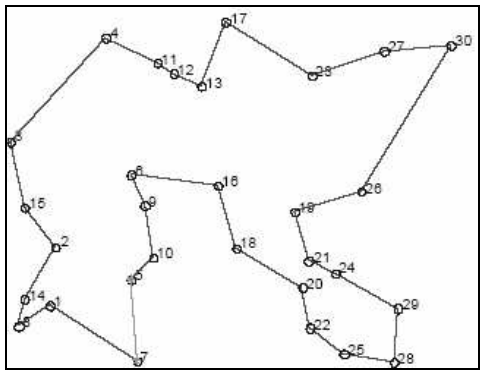


Figure 6 - The best tour found after 300 generations, taken from [4]

Again, CLONALG was configured for function optimisation where a set of the higher affinity clones become candidates to the memory antibody pool, rather than a single individual. The population size (N) used was 300, and the total new antibodies inserted (d) was 60, though this occurred very 20 generations to give the system an additional

chance to explore. The clonal rate (β) used was two and the number of antibodies selected each generation was half the population size – 150.

4.3 Parallel CLONALG

The immune system is inherently parallel system both in terms of distributed nature of its control, the distributed nature of the response and maturation process and the management of the cell population. Work by Watkins, Bi and Phadke [8] investigated augmenting the CLONALG technique to be more efficient by operating in parallel. This was achieved by partitioning the work to be completed by the algorithm, that is the antigen set among the number of available processes. The task of each process is then to prepare an independent memory antigen pool. On the completion of all tasks, a root process then collects all prepared memory pools which are combined or merged into a master memory pool.

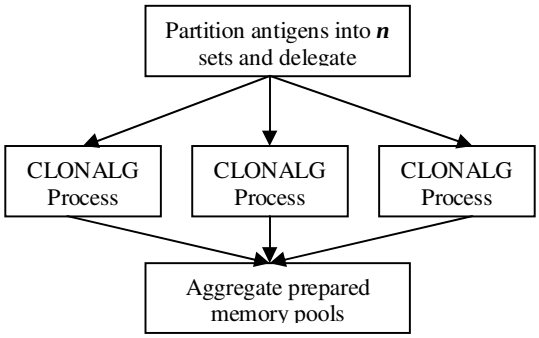


Figure 7 - Schematic overview of the parallel version of CLONALG

The parallel version of CLONALG was tested on the binary character recognition problem previous used in [4,5]. Tests were performed varying three critical parameters of the system, namely the total antibodies (N), the number of selected highest affinity antibodies (n), and the number of new antibodies that replace low affinity antibodies in the remainder antibody pool (d). A speedup metric was proposed that divided the time taken to complete the problem using serial CLONALG, divided by the time taken by parallel CLONALG. It was proposed that the speedup expected from parallelising CLONALG would be linear, though interestingly, the results showed a super-linear relationship given the number of processes tested.

4.4 CLONCLAS

Work by White and Garrett [7] extended work on applying CLONALG to the problem of pattern recognition problem, and investigated the performance on the technique on classifying unseen character instances. An augmented version of CLONALG was proposed called CLONCLAS (CLONal selection algorithm for CLASsification). Each antibody in the population was allocated a class, thus allowing it to perform classification by assigning its class to an antigen. When an unseen antigen is exposed to the population, it is allocated the class of the antibody with the highest affinity to said antigen. An affinity threshold parameter was introduced which defined the sphere (or hyper-sphere)

influence of a given antigen in the affinity landscape, allowing bounded partial matches or approximations to be used for the classification task.

Although the CLONCLAS technique was indicated to be designed for classification, the technique was tested only on the previous mentioned binary pattern recognition dataset. The augmentation required that each class (each character) to be allocated a single exemplar, and that the system would be able to generalise from multiple samples using this single data point. Classification of unseen character instances is performed using the prepared memory pool. The antibody with the best match to the antigen is selected, then if the affinity is less than the threshold, the antigen is classified using the antibodies class, otherwise the class of unknown is allocated.

An additional augmentation was made to CLONALG to form the CLONCLAS technique. It was indicated that the CLONALG technique does not take advantage of the temporary knowledge gained through the cloning and affinity maturation process. It was proposed that CLONCLAS would capitalise on this information, by using remaining high-affinity clones to speed up the algorithm. An additional step was added to the procedure after candidate memory antibodies are entertained for entry into the pool. This new step required that the remainder antibody pool be cleared and populated with the same number of high-affinity antibodies from the cloning and affinity maturation process. After this re-population, the standard replacement with new antibodies process would occur.

Both the CLONALG and CLONCLAS techniques were tested against four other naïve Hamming distance based techniques on three problem domains. These problems consisted of the original binary character pattern recognition dataset, a version of the dataset where additional characters were added in different font and normalised to the correct size, and finally the same set, where the additional characters were not resized.

The score for each algorithm was taken as the average over 10 runs. The techniques were tested over a range of different generations, though the remaining parameters used for CLONALG and CLONCLAS were the same as those used in the original implementation of the problem in [5]. For the first task, CLONCLAS was able to achieve accuracy of 87.5% in one tenth the number of generations of CLONALG (50, compared to 500). Both techniques performed equal well on the second task, though were outperformed by one of the naïve Hamming distance based techniques. Finally, on the last task, both techniques performed poorly, though it should be noted that the final task was of a higher level of difficulty compared to the first two.

A discussion of the results indicated that the augmentations made to derive CLONCLAS mean that the two techniques work slightly differently to achieve their goal. CLONALG was seen to perform a random global search, and then use mutation to perform a refining local search, where as CLONCLAS was seen proceed via a directed random walk through the domain. The proposed CLONCLAS was show to provide a slight performance increase for seen examples, though the same performance for unseen examples, indicating that perhaps it over learned the data it was exposed to. Further

research into the technique was recommended, specifically looking at areas such as dynamic stopping criterion and the sensitivity of user parameters.

4.5 Dynamic Function Optimisation

The link between CLONALG and Evolutionary Algorithms (EA's) and specifically Evolution Strategies (ES's) were originally raised in [4], and further expanded upon in [5]. It was indicated that given that CLONALG implements the three main elements observed in evolution, namely diversity, selection and variation, that it could be considered an evolutionary algorithm, given the clear use of immunological rather than evolutionary vocabulary. The parallels with evolutionary search techniques are made clearer when considering that CLONALG operates primarily using cumulative blind variation and selection.

Work in [5] compared the manner in which CLONAL maintains a diverse population and searches in parallel in similar to that of niching genetic algorithms such as fitness sharing. Given the similarity, the two techniques were compared on the multimodal test problems, where CLONALG was shown to achieve better grouping or clustering around area of interest in the search space (extrema of the function). Given the clear relationship between the two biologically inspired techniques, work in [9] investigated and compared CLONALG and ES when applied to a dynamic function optimisation domain.

Evolution strategy are a type of evolutionary algorithm that use real-valued vectors and operate using selection and variation via mutation [10]. The mutation scheme applied typically involves adjusting real vector values using a Gaussian distribution and a self-adaptation rule for automatically determining the best mutation amount or evolution window. A test bed of two very different dynamic function optimisation problems were provided (peaks and circles), and the problems were tested over two, five and 10 dimensions. The algorithms were executed on the problems for approximately 1000 function optimisation then the optima was randomly changed. This was repeated 20 times so that each test involved close to $(20 \cdot 1000)$ function evaluations. A barrage of population parameter sensitivity tests were performed for both CLONALG and ES across the set of problem domains. Parameters investigated included the ratio of parents to offspring each iteration as well as the number of newly inserted random immigrants. Both approach were shown to be reasonably sensitive to the parameters, particularly the number of immigrants inserted each iteration.

Both the ES and CS (Clonal Selection) techniques were compared. Interestingly CLONALG was shown to be slower at optimising, though achieved the better results for the two-dimensional versions of the dynamic functions. The ES achieved the better results for the eight and 10 dimension version of the peak and circles problems. It was unclear whether the ES's inability to achieve better results on the two-dimensional problem was caused by its inability to adapt to change or the techniques lack of or premature convergence. The final populations of both techniques were plotted for the two-dimensional problem, and it was shown that CLONALG completed with a more clustered population, where as the ES was more diverse, spread across the domain.

4.6 Adaptive Clonal Selection (ACS)

Further capitalising on the close relationship between ES and CLONALG, work by Garrett [11] identified a number of shortcomings of the CLONALG technique, and in addressing them, devised a new algorithm called Adaptive Clonal Selection (ACS) that represented a parameter free version of CLONALG. The parameter-less technique was devised for and applied to static function optimisation problem domains with success. During the development of ACS, the following shortcomings were identified of the CLONALG technique:

1. The technique can be costly in terms of the number of function evaluations required
2. Uses a binary antibody representation for solutions which was claimed to be limited in its precision
3. The technique is not adaptive to variations in the function response surface
4. Finally, CLONALG has a number of user-defined parameters that typically require a sensitivity analysis to find good configuration

A number of specific inefficiencies with CLONALG were also identified as follows:

Defining Functions – it is claimed that the technique requires the user to select functions for determining the number of clones to produce and the degree of mutation. Further, this can require additional parameters for scaling the number of clones (β), and the amount of mutation (p)

Defining *n* – selecting the number of high-affinity antibodies for cloning and maturation is non-trivial and has a strong effect on algorithm performance

Too many evaluations – the technique has two selection and evaluation stages, which may not be necessary

Scalability – The number of clones created each generation, and thus the number of function evaluations is typically larger than N, this is expected to become unmanageable for larger population sizes

To address the identified shortcomings and inefficiencies with CLONALG, the following changes were both proposed and implemented in what became the ACS technique:

1. Remove identified inefficiencies. This was achieved by evaluating the antibody pool once after initialisation, and evaluate clones once after creation. The assigned affinity values stay with the antibody for its duration and thus do not need to be recalculated.
2. Algorithm runs for a defined number of function evaluations (stopping criterion). Achieved by configuring the system to stop execution after 1000 function evaluations.
3. Use real-valued antibodies rather than a binary string representation.

4. Remove the p parameter used in the used mutation function. An ES based mutation scheme was employed where each antibody had its own mutation rate which was divided by multiplied by the magic ES value of 1.3 each generation.
5. Remove the n parameter used to select a set of high-affinity antibodies, and the β parameter used to scale the number of clones produced. This was achieved by using an ES based mutation scheme for both parameters similar to antibody mutation. The initial value of one was applied to each parameter during the initialisation stage, and the ES manipulation of the parameters values occurred at the end of each algorithm iteration.

The proposed algorithm was also implemented slightly differently from the canonical CLONALG definition from [5]. This changed involved removing the concept of the remainder population and thus allowing new random antibodies to be inserted (not replaced) into the main antibody population. The replacement of low affinity members from the remainder pool step from CLONALG was replaced with a cell-death step in the procedure where low-affinity antibodies are killed (deleted) from the pool until its size reaches N. The following figure provides a simply schematic overview of the proposed ACS technique:

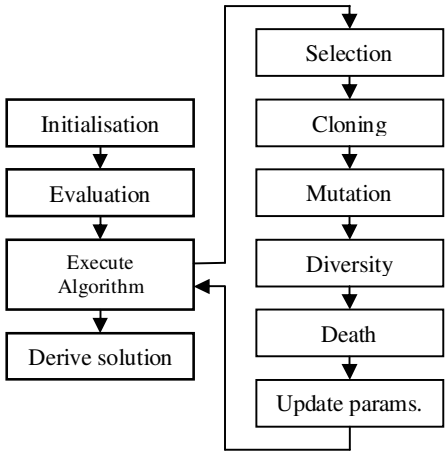


Figure 8 - Overview of the ACS algorithm

The proposed ACS technique was tested on four multimodal function optimisation problems on a range of dimensions (1, 2, 4, 8, 16 and 32). Five different variants of the technique were evaluated and compared to a standard random solution generator rather than the standard CLONALG technique. The results indicated that ACS performed well and the ES-like automatic discovery of parameters were shown to be broadly similar to those found through off-line parameter sensitivity analysis. Although a few parameters remained (population size N, number of function evaluations, number of random insertions d, and initial values for β and n), that dynamic parameter adjustment in CLONALG is a worthy area for investigation. It also showed that real-valued antibodies and the use of ES-like mutation schemes on antibodies might be a useful addition to the technique.

5. CLONALG for Classification

Classification is a class of problem where an algorithm is exposed to feature vectors from a domain and assigns a known class label. The more common classification algorithms learn in by what's called supervised learning where a set of training examples that include example feature vectors and known classification labels are used to prepare the system. Strictly speaking, the CLONALG technique has not been applied to classification problems in this manner (to the author's knowledge). The closest application of CLONALG was the CLONCLAS variation of CLONALG that provided a classification-like implementation applied to a binary character pattern recognition problem domain.

Given the lack of confidence in a naïve application of CLONALG to classification based on the results of CLONCLAS, the intent of this section is to propose a new implementation of the clonal selection theory for classification problem domains. This new algorithm called the Clonal Selection Classification algorithm (CSCA), and is based on desirable features observed of CLONALG and its variants. The proposed algorithm takes advantage of the knowledge of previous implementations of the technique to other domains and attempts to apply that knowledge to design and devise an effective, easy to use clonal selection inspired classification algorithm. It should be noted that a naïve CLONALG implementation for classification and the CSCA algorithm were implemented as plug-ins for the WEKA machine learning workbench [12], see section 8. for further details.

5.1 Abstracted CLONALG

CLONALG is already an abstraction of the clonal selection theory, devised to addressing complex engineering applications. The features and functioning of the technique have already been discussed at length. To devise a new implementation of the clonal selection for classification domains requires a speculation and even an identification of CLONALGS source of power. Like an evolutionary algorithm, CLONALG operates through the simple process of variation and selection, and like niching techniques has the added capability of maintenance of diversity. Thus, CLONALG can be abstracted to the following high-level functional units that describe its behaviour:

1. **Population Based** – The system is population based, meaning all knowledge the technique gains through sampling the domain is stored by those samples in the population. This allows new information to replace old disused or irrelevant information, it allows an abstraction or generalisation of information, though it also allows loss of useful information
2. **Selection** – Selection, a critical element of evolutionary-inspired systems has also been shown to be a critical element of the Clonal Selection Theory of acquired immunity. In both cases, it provides the bind directedness used in the search performed in most problem spaces.
3. **Sample Generation** – The traditional concern with sample generation is the trade-off between exploration of the domain (the new), and exploitation of the acquired knowledge (the old or the known in the population). The technique has an explicit version of each in both the cloning-maturation process and the random sample insertion process.

It would be a valid argument to say that CLONALG is simply an evolutionary algorithm dressed up in immunological terminology, a point briefly already addressed by de Castro and Von Zuben [4,5]. Given the gross simplification of the techniques function described above, this is quite obvious, therefore it is important to outline those points that differentiate the two biological inspired search techniques.

This is a sticky topic that seems to be somewhat held at a distance in the field of artificial immune systems. As far as the author can determine the single distinct difference is in fact the terminology and metaphor. This is a point already mentioned though is the basis for the techniques existence nonetheless. The fact that the metaphors are different, yet fundamentally similar is interesting. It could be argued that the closer an algorithm implementation came to its selected metaphor, the more different the techniques become. This is the critical element of this argument. As has already been observed, the further an implementation diverges from its metaphor and borrows from other metaphors, the more the distinction between techniques blurs, (ACS is an example). Another important point is that fact that the topic for discussion is algorithms for addressing complex problems and not metaphor simulations, thus devising a technique that matches a metaphor too closely is likely to miss the point and result in poor performance.

Clearly, some balance is required. Biological metaphors are used for the basis for computer algorithms because they exhibit desirable features for solving problems. From a practical standpoint, the terminology or field from which a technique is derived is of little consequence, what matters is the techniques usefulness (results, easy of use, etc.). This rational will be the basis for the development of clonal selection-inspired classification technique, though it is obvious that this is an area in strong need of exploration – for the reuse of parallel ideas between fields at the very least.

5.2 Desirable Features

The CLONCLAS and ACS techniques provide a strong foundation for implementing CLONALG for classification given their departure from the standard CLONALG implementation, though their poor results and different application respectively dissuade a naïve implementation of CLONALG to classification.. This section lists and describes those features from previous CLONALG implementations that are desirable for a new clonal selection inspired classification system

5.2.1 Canonical CLONALG

The canonical CLONALG technique exhibits a number of desirable features both in terms of search capability and distinction between evolutionary and immunological approaches. These desirable features included:

Deterministic – CLONALG is deterministic, in terms of selection, cloning, affinity maturation and antibody replacement, a feature unlike most evolutionary algorithms which are probabilistic. This feature is desirable both because it aids in the differentiation between the techniques, and provides predictability and visibility of the decisions and processes that occur within the algorithm. Further, this point is a practical consideration

where the algorithm practitioner requires a strong understanding to both to be able to explain the system outcomes and tune system performance predictably.

Affinity-based sample generation – An algorithm feature inspired by the clonal selection theory that seems to be departed from in some CLONALG applications is the affinity basis for controlling the amount of local exploration performed from selected antibodies. This feature is desirable because it is a loose approximation of the base metaphor and provides an automatically regulated search focus across the problem domain.

Random sample introduction – A simple, yet apparently effective concept of periodic random sample introduction to the system provides a potential means of escaping local optima. For classification, it provides a means of introducing completely new proto-exemplars that may or may not be useful to the training process.

5.2.2 CLONCLAS

As mentioned, CLONCLAS performed a classification-like task for a pattern recognition domain, though it was shown to perform relatively poorly. Given the similarity to classification problem domain, it provides a useful starting point.

Generalisation – CLONCLAS described the need for generalisation through treating each antibody as an exemplar. Although one antibody was maintained per-class (not practical for most classification problems), this generalisation feature is desired from the proposed classifier, and is the basis used in the AIRS (Artificial Immune Recognition System) classification technique [6].

Class assignment – in the technique, the training dataset is taken as the antibody population, and it becomes the role of the system to generalise the characteristics of members from each class into class-exemplars used for classifying unseen examples. CLONCLAS used an affinity threshold which, although a useful concept for bounding the influence of exemplars, requires an additional parameter to be specified. Classification of unseen examples will be performed using an affinity-competitive environment, and could even be implemented using a majority vote k-Nearest Neighbour approach, the same as used in AIRS. The threshold is required of course, though can be just as easily achieved naturally using a competitive population of antibodies.

5.2.3 ACS

The ACS technique demonstrated a number of desirable augmentations to CLONALG. Some of those features desirable for a new clonal selection technique include:

Natural-valued vectors – more than using simply real-values to represent characteristics, it is desirable to use a natural attribute representation. This would allow nominal, real, discrete and other attribute types to be handled correctly as long as a comparison of some kind can be made between values.

Simplified implementation – ACS was described as using a single antibody pool (no distinction between memory and remainder pools). This simplifies the implementation with no obvious detrimental effect to the core technique. Like the ACS implementation, it requires that some population control mechanism be employed when inserting recognition units (antibodies) into the population such as affinity-based replacement or affinity-based deletion. Also provided in this simplified implementation was the removal of inefficiencies such as only evaluating the affinity of antibodies once. Although antibodies have to be evaluated each time a training instance is observed, these affinities could be cached internally by the antibody and used once.

Minimal user parameters – A design goal of ACS was the removal or minimisation of user parameter. This is a desirable feature, though more so if the user-defined parameters of the system that remain are obvious in their effect on the system, and relatively insensitive across ranges. This would allow a general configuration to still achieve a near-optimal result.

Adaptable – The adaptability observed in ACS is desirable both in terms of the removal of configuration variables and efficiency of training time. This feature is desirable for those suitable algorithm elements that can be meaningfully automatically discovered – meaning a careful investigation and analysis as justification.

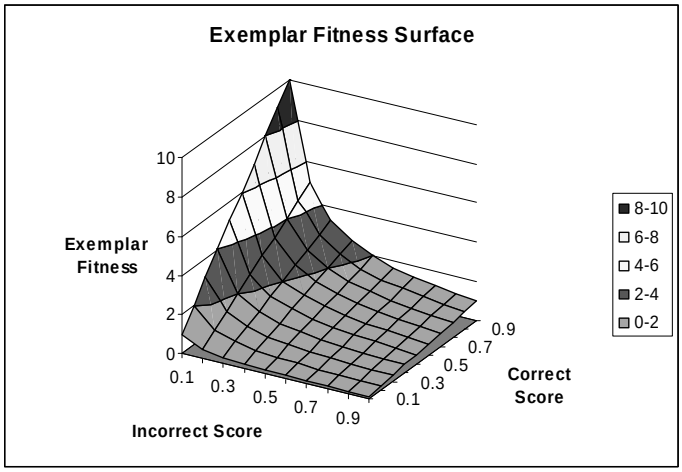
5.3 Clonal Selection Classifier Algorithm (CSCA)

The goal of a classification exemplar is simply to maximise classification accuracy and minimise misclassification accuracy. This can be expressed as an exemplar fitness function as follows:

$$\max\left(\frac{correct}{incorrect}\right)$$

Equation 4 - Exemplar fitness function

This simply equation produces a surface as follows:



Equation 5 - Exemplar fitness surface, correct and incorrect scores between [0.1, 1.0]

This surface exhibits useful properties for the intended purpose as an exemplar evaluation function. When taken as a maximisation function, the system seeks to achieve the desired effect mentioned above where correct scores are maximise and incorrect scores are minimised. Used in conjunction with elements of clonal selection, this transforms the problem of classification into that of function optimisation, though unlike function optimisation problem domains, the system seeks to discover a diverse population of high fitness exemplars (antibodies), rather than a single antibody solution. Further, the surface portrayed in the above figure is representative of the surface observed during training in that the x and y axis are of the same scale and bounds as each other meaning a known bounds to the function which can be inspected for a give problem domain.

This simple idea of representing a classification problem as that of function optimisation is a departure from the CLONALG technique. All that is required to allocate fitness is an evaluation of the number of correctly and incorrectly classified training instances. This can only be achieved by exposing the antibody population to a set of antigens, and requiring each antibody to keep score. The problem with this proposition is that some datasets are extensively large, or unavailable as a single unit at any one time, and thus treating a single exposure to all antigens as a training iteration (generation) is not entirely practical. This is easy overcome by using the epoch approach to training from the artificial neural network field of study. This is where the training set is broken into *p* partitions, and a single algorithm iteration or epoch is taken as the systems exposure to a single training set partition. It should be pointed out that for the majority, if not all standard (toy) machine learning classification problems, this is not a concern.

This batch training methodology is reminiscent of AIRS, though here, the system is permitted multiple exposures to the antigen set, rather than a single exposure. This approach is also reminiscent of classification neural networks such as Learning Vector Quantisation (LVQ), in that it allows the system to adapt the population to the training set in one go, or at any discrete level (epoch size), rather than forcing the system to update in an online manner (singe antigen at a time) as in the CLONALG technique.

Once exposure to the antigen (more specifically an antigen set partition) has occurred, the system must adapt based on the allocated antibody fitness. The fitness is metaphoric for affinity with an antigen exposure or dose of antigens such as in the case of an inoculation with a vaccine. More accurately, it could be said that the inoculation consists of multiple vaccinations, where each class represents an independent vaccine, and data instances of the class are representative of the unique antigens/antibodies (units) of that vaccine. The important point from this abstraction is that each unit introduced to the system is unique, and thus each recognition unit of the system will have a unique specificity or affinity with each member of the training set. This affinity (distance measure) could be accumulated by each antibody and used as a fine grain correct / incorrect tally in the exemplar fitness equation.

A few simple rules can be applied to pre-process the antibody population before cloning and maturation as follows:

1. Those antibodies with zero correct, and more than zero incorrect counts are switched to the class of those with most hits, and fitness is recalculated
2. Those antibodies that have a fitness below a defined threshold (ε) are removed from the population and do not participate in cloning and maturation
3. Those antibodies with zero misclassification do not participate in cloning and maturation

These rules ensure that those antibodies that effective are maintained by the system (long lived memory cells and promotion of effective units), and those that are unused are removed to make space for more useful units (cell death). The remaining antibodies than participate in cloning and affinity maturation (sample generation) in negative selection manner.

Cloning and maturation, the amount of local exploration is based on an antibodies is relative to their usefulness. A simple resource limitation scheme is required for practical reasons where the total number of varied clones produced equals the number of antigens in the epoch (size of p). This number could quite easily be a user-defined parameter or some other system component, though the size of the epoch was chosen because it is assumed that if the system can maintain the n antigens, then it can maintain n additional antibodies in addition to the antibody pool of the system. This value could also be adjusted using a scale factor to either increase or decrease the number of clones produced each generation.

The "selected" (negatively selected) antibodies are cloned in relation their fitness to ensure that those with poor fitness receive the most attention. The number of clones for each member of the selected set is taken as:

$$numClones_i = \left(\frac{f_i}{\sum_{j=1}^{S} f_j} \right) \cdot (n \cdot \alpha)$$

Equation 6 - Calculation for the number of clones produced for an antibody

where f_i is the fitness of the antibody in question, S is the total number of antibodies in the selected set, n is the total number of antigens in the current epoch, and α is an optional scale factor that defaults at one.

Each clone is then mutated using a similar policy as the one used to determine the number of clones. In this case, the fitness ratio of an antibody is used to determine the range on each attribute that a clone can take. This assumes that the bounds of each attribute is known or can be reasonably estimated for the domain in question. Using the same fitness ratio used to calculate the number of clones to produce, the mutation range of each attribute is given as:

$$r_i = ratio \cdot range_i$$

Equation 7 - Range a mutated clones attribute can be within

where the centre of r_i is the attributes current value a_i, and the *range* is the known or estimated range of the attributes possible values. New attribute values are then calculated as a random value within the defined range r_i. This is only used for real values; nominal values simply have a random nominal value selected from the range of known nominal values. Rather than taking a uniformly distributed random number, it may be more useful to use another probability distribution function such as Gaussian.

A feature of the proposed algorithm is that it provides the ability to automatically discover (evolve) an appropriate antibody population size. All maturated clones prepared are added to the antibody population and permitted the opportunity to compete for survival through their usefulness on the training set (competitive learning). A point not discussed is that although selection and fitness (affinity or usefulness) define the population size, it is possible for the algorithm to bloat. This means that the algorithm needs to protect itself form simply evolving an antibody population that duplicates the antigen population. This is achieved in two ways:

A limit on the number of generations (*G*) – Limiting the number of times the system is exposed to the antigens, allows the user to define the amount of adaptation the system should perform for a given problem domain, and provides a stopping criterion for the technique. The primary concern with this user-defined parameter is the setting of an appropriate term that provides sufficient learning, though prevents the system from over-learning the training data. A generation is taken as a single iteration of the algorithm, though more accurately, when the training dataset is partitioned a single generation is taken as the systems exposure to a single epoch or partition of training instances.

The minimum fitness threshold (ε) – Provides a basic population control mechanism that removes or deletes those cells that are not useful to the system. The benefit of this user-defined parameter is that its effect is predictable and obvious on a given population of antibodies. Further, it is possible to adjust this parameter dynamically during a run based on an evaluation of antibody population size and the number of deletions per generation.

The final step of the algorithms procedure is to “top-up” the population by inserting new antibodies. It is not expected for this top-up to provide extremely useful antibodies each generation, though as mentioned, it may provide the capability for the system to escape a premature convergence to a suboptimal set of classification antibodies. New antibodies can be either generated randomly within the bounds of each feature or attribute, or simply taken as randomly selected antigen instances. The total number of new antibodies inserted into the population each generation is the precise number of antibodies selected (negatively selected) for cloning. The reason for this is that the number of selected antibodies each generation is expected to be low, and the maturation process is expected

to produce new antibodies that eventually either replace the selected set or consolidate the selected set. This allows the number of new antibodies inserted each generation to adapt with the amount of selection performed each generation. It is possible and may be useful to have this parameter either user defined or adapted by some other means.

5.4 CSCA Specification

This section discusses implementation details of the proposed Clonal Selection Classification Algorithm (CSCA). Provided are discussions on the user-defined parameters, pseudo-code and schematic representations of the algorithm and discussions on potential alternative or optional elements of the procedure to implement.

The user-defined parameters and prior knowledge of the algorithm are as follows:

Name	Required / Optional	Description
Attribute bounds	Required	The bounds of each attribute must be specified before execution of the algorithm, and can be either actual or approximated. This bounds is only required for training, not production use of the technique. In the case of nominal attributes, the range of possible values is required.
Initial population size (S)	Required	Defines the number of antigens with which to seed the antibody population, where $S \in (0, p]$ and p is the total partition size. Defaults to 50.
Total generations (G)	Required	The total number of generations for which to execute the training scheme. A problem dependent variable. Defaults to less than 10, example: five.
Random number seed (r)	Required	The value with which to seed the random number generator, a parameter common to all algorithms with a stochastic component. Provided for consistency and repeatability of results. It is common to use the system time in milliseconds when these issues are not of concern. Defaults to one.
Clonal scale factor (α)	*Optional*	Used to either increase or decrease the number of clones produced each generation. Defaults to one.
Minimum fitness threshold (ε)	*Optional*	Used to prune control the antibody population size. Those antibodies with a fitness allocation $\leq$ this threshold are deleted from the pool. Defaults to one.
k-Nearest Neighbours	*Optional*	Used during classification to select the best k matches to a given unclassified pattern. Classification then occurs by selecting the majority class from the k selected antibodies, or the first class index in the event of a tie. Defaults to one.
Partition size (p)	*Optional*	Used for very large datasets and online learning, defines the number of training instances exposed per the system per generation. Defaults to one.

Table 1 - Overview of user-defined parameters and prior knowledge for CSCA

The following provides a schematic overview of the discrete functional units of the SCSA technique:

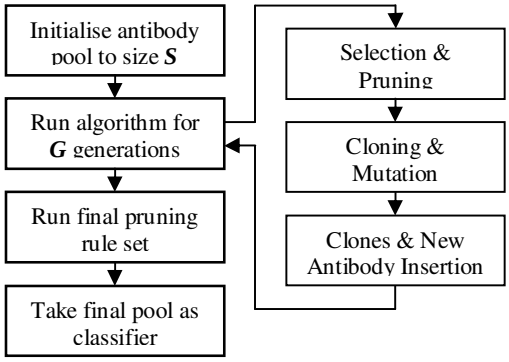


Figure 9 - Overview of the CSCA technique

The following provides a detailed specification of the main steps of CSCA:

1. **Initialisation** – Involves populating the antibody pool with ***S*** randomly selected antigens (without replacement during initialisation).
2. **Loop** – Involves running the main steps of the algorithm for ***G*** generations.
    a. **Selection and Pruning** – Involves exposing the entire population to the antigen set and calculating fitness scores for each antibody. The entire population is then selected. Selection rules are then applied in the following order
        i. Those antibodies with a misclassification score of zero are removed from the selected set
        ii. Those antibodies with zero correct classifications and > zero misclassifications have their class adjusted to the class with the majority, or the first class indeed in the case of a tie. Fitness is then recalculated
        iii. Those antibodies with a fitness scoring below (***ε***) are deleted from the selected set and the base antibody population
    b. **Cloning and Mutation** – The selected set is then cloned and mutated using fitness proportionate operations
    c. **Insertion** – Generated clones are inserted into the main antibody population. *n* randomly selected antigens from the antigen set are inserted into the population, where *n* is the number of antibodies in the selective set from step 2.a.
3. **Final Pruning** – Involves preparing the antibody population for production application. The population is exposed to the antigen population a final time, fitness scores are prepared, and pruning is performed as in step 2.a.iii.
4. **Classification** – The antibody population is taken as a set of exemplars. For a given unclassified data instance, it is exposed to the population. The ***k*** best matches (highest affinity) are selected, and the majority vote for class of the antigens is applied to the unclassified instance.

The following list additional user-configurable and optional elements of the algorithms implementation. Specifically, this section provides a number of potential areas for

alternative implementation and augmentation that may be suitable for further investigation. This section can be considered a list of potential specialisations of the general CSCA technique.

Affinity – Affinity is taken as the distance between a given antibody and antigen. Specifically it is the Euclidean distance between the vectors represented by an exemplar of the system and a training instance. In the case of a nominal value, the attribute distance is taken as a Boolean match (0) no match (1). Other attribute specific distance measures can be used such as temporal difference for date attributes and edit difference for string attributes or Hamming distance is bit strings are used.

Partitioning of training set – As mentioned, the training dataset can be partitioned for exposure to the system. This is a simple addition to the CSCA technique that can be used in the case of very large training datasets, and in the case of online-learning. If applied, it requires an additional user parameter $\boldsymbol{p}$, which defines the size of a partition. For very large datasets, a partition can be taken as p randomly chosen data instances (without reselection until all partitions are exposed to the system), or as the entire dataset divided into p equal blocks. In the case of online learning, p can be used as the number of units to gather before training the system. Finally, the partition approach can be used to execute the CSCA algorithm in parallel in the same manner as was used for parallel CLONALG [8], where p represents the total number of algorithm processes.

Number of Generations – The number of generations used to train the system can be speculated at based on the algorithms performance on other similar classification problems. A more effective strategy may be to perform some form of sensitivity analysis for a given value. It is expected that more intelligent approaches can be developed, including using something such as change in mean quantisation error between antibodies and antigens or change in fitness to indicate when an appropriate time for training to complete.

Assigning classification scores – As mentioned, alternative means for allocating responsibility for error may be used besides a tally of correct and incorrect classifications. One alternative already speculated at was using accumulated stimulation values. Another approach that is compatible with the tally and accumulation approach is to select k-nearest neighbours to an antigen and allocate responsibility to all k antibodies. A final potentially interesting aspect of this component may be to allow antibodies to dynamically select the class label to which they belong. This could be achieved by allowing instances to always select the class with the highest tally or score at the end of each generation (similar to the class switch behaviour of the system).

Selection rules – As shown, the selection process is actually a form of negative selection, where the process assumes all antibodies will participate in cloning and maturation, and using a rule set to manipulate and prune the selected set. The rules applied are simplistic, though hold true to the intent of the technique. It is quite possible to extend the rule set to further prune more or less antibodies from the set. Perhaps an upper-stimulation threshold

could be applied to determine the amount of misclassification an antibody must have to be considered in need of cloning and maturation.

Replacement strategy – Generated clones and new antibodies do not replace population members directly, rather they are inserted into the population then the population as a whole is reshaped using ε. An alternative approach to antibody insertion is to use some form of affinity based replacement strategy as is used on CLONALG. This may involve simply replacing the lower fitness members and thus allowing ε to be removed, or may involve some form of affinity matching policy between clone antibodies and members of the antibody pool.

5.5 CSCA Statistics

One of the design goals of the CLONALG based classification system was visualisation of the system, meaning that the system provides feedback and insight into its functioning to the user. One useful means of achieving this is by providing running statistics of the algorithm. This section lists a number of potentially useful statistics that can be reported during the running of CSCA.

5.5.1 Training Statistics

These are statistics that can be gathered and potted at runtime during the training schedule of the algorithm. Alternatively these statistics can be gathered at the end of training and provided as a training report. (Note; when a mean scoring is taken, it is assumed that the standard deviation is also provided).

Mean antibodies pruned per generation – The average number of antibodies pruned from population (fitness $\leq \varepsilon$) per algorithm generation. Provides an indication of the effectiveness of the ε user parameter, and the amount of selective pressure the population undergoes during training.

Mean population size per generation – The average size of the entire antibody pool at the end of each generation. Provides an indication whether the ε parameter needs to be increased or decreased.

Mean antibodies without misclassification per generation – The average number of antibodies removed from the selection set due to having zero misclassifications, per generation. Provides an indication of the state of the system and its convergence to a usable classifier.

Mean classification accuracy on training data per generation – Provides an indication of the evolving classifiers performance on the training data. May be useful as an indication of over-learning – a time when the training scheme should be halted.

Mean antibody fitness per generation – The fitness of all evaluated antibodies each generation. Provides an indication if the ε user parameter may need to be raised or lowered. Also provides some indication of the state of the classifier over time. This

statistic may be useful in conjunction with the minimum and maximum fitness each generation.

Mean antibody class switches per generation – The average number of antibodies that switch classes each generation. Given that the antibody class cannot change by any other means (such as mutation), the amount of intelligent class switching each generation may provide an indication of the system correcting bad training data or shaping class decision boundaries. Less class switching is expected towards the end of the training schedule, and this may be used as an additional piece of information in an early-stopping criterion.

Mean new antigens per generation – The average number of training instances that are converted to antibodies and inserted each generation (alternatively could be randomly generated data instances). Provides an indication of the degree of diversity injections occurring over the course of training. This statistic would be useful if extended to provide information on the benefit of injected antibodies, such as mean immigrant existence duration or contribution to overall accuracy.

Mean selection set size per generation – The average size of the selection set each generation. Provides an indication of the size of population segment participating in the search for improved exemplars. A lower percentage of the population being selected indicates either too much antibody pruning, or a system that is over-learning the training data.

5.5.2 Classifier Statistics

The statistics provided in this section are intended for use after the completion of the SCSA training scheme. They are excepted to provide some insight into the nature of the classifier produced by the algorithm.

Classifier size – The number of antibodies in the classifier provided. It is expected that the number of antibodies is less than the number of antigens used to train the system. If this is not the case, it may be better to simply use the training dataset in a k-nearest neighbour algorithm to classify unseen data instances.

Class breakdown – A breakdown of the number of antibodies that belong to each of the known classes in the training set. A disproportionate distribution may indicate special attention to a class or area of the problem domain given increased difficulty or over representation in the training dataset.

Data reduction – The amount of data reduction (generalisation) the classifier was able to achieve from the training dataset. For a classifier to be a useful, it is expected to have less antibodies than there are in the training dataset, this can be represented as a reduction percentage.

5.6 CSCA Preliminary Analysis

The proposed SCSA technique has not yet been rigorously tested. An initial implementation of the technique has been devised as a plug-in for the WEKA machine

learning workbench (see section 8.). It should be noted that the implementation included all of the CSCA statistics specified in section 5.5, which were used to aid in preliminary analysis the proposed algorithm. Ad hoc testing was performed on a number of standard classification problems from UCI machine learning repository [13], specifically Iris Plants, Pima Indian diabetes, Sonar, Wisconsin Breast Cancer, Cleveland Heart Disease, and Balance Scale Weight & Distance. Datasets were selected that consisted of a mixture of numeric, discrete and nominal attributes, and results using default algorithm parameters were very encouraging. For some of the selected datasets, results were competitive with the top 10 best known classifiers for the problem, listed on [14,15].

Although a sensitivity analysis has not been performed on the user parameters of CSCA, initial testing revealed an inherent robustness in the technique. Variations of the initial population size (S) seemed to have little if any impact on the resulting classifiers performance, likely given the automatic self-regulation of the technique. The number of training generations (G) was also seen to be robust, achieving good or close to good results with values ≤ 10. The clonal scaling factor (α) was seen to be effective at its default value of one. Values less than one resulted in lower classification accuracy on most problems, likely caused by the decreased searching capability of the system.

Higher k values resulted in slightly better performance on some problems, though a value of one was shown to be useful across the board. A magic value of one for the minimum pruning fitness (ε) parameter was shown to useful for all of the tested problem domains. Values of zero resulted in a classifier with similar classification accuracy, though far lower data reduction. Values > one were showed increased data reduction, and improved results on a small number of the tested problems. Adjustments to the number of partitions parameter (p) resulted in a classifier that showed significant increases in data reduction, though executed faster for all problems and performed better for some problem domains.

The system exhibited excellent data reduction with default parameters. Data reduction scores > 50% were observed on all problems, in some cases, scores as high as 80% to 88% were observed whilst still achieving competitive classification accuracy. It is speculated that the minimum pruning fitness parameter controls the amount of selective pressure in the population, and this in conjunction with the exemplar-based fitness allocation and selection rule set allow the technique to rapidly and beneficially generalise from the training set.

Given preliminary testing, a number of minor algorithm implementation details were adjusted and found to improve the quality of the procedure, these included:

1. Automatic stopping condition – In the case when the algorithm has an empty selection set, the procedure will exit. This means that the population only consists of antibodies with zero misclassifications and has likely over learned the training data
2. Lax pruning procedure when using partitions – In the case were multiple partitions are being used, the system will prune those antibodies that are useful in one partition and not useful in another. This requires that the pruning (minimum

pruning fitness) be lowered to zero when partitions are used, then increased to the user-defined parameter for the final pruning stage.

The following list summarises the observed benefits of the technique from preliminary testing:

1. Apparent lack of sensitivity to required parameters, specifically the initial population size, number of generations, kNN
2. Apparent suitability of default parameter values on a range of problems, specifically clonal scale factor and minimum pruning fitness of one
3. Apparent quality of classifiers produced across a range of standard datasets that included numeric, discrete and nominal attributes
4. Apparent excellent data reduction capability across a range of standard datasets

In addition, the techniques implementation offers the following features:

1. Competitive learning between antibodies in the population
2. Meaningful Individual antibody affinity (fitness criterion) for classification problems
3. Self-regulation of total resources required for classifier on a given problem
4. Extensible in that the procedure is naturally broken down into modular discrete function units that can be rearranged and replaced as required

6. Further Work

The proposed Clonal Selection Classification Algorithm (CSCA) represents a new classification algorithm based on abstractions of the clonal selection theory of acquired immunity and the CLONALG technique and its variations. Given the success observed during preliminary testing, the proposed algorithm warrants, nay demands further investigation. This investigation would be expected to include some, if not all of the following tasks:

1. Rigorous empirical evaluation of the technique across a broad range of common machine learning datasets
2. Rigorous sensitivity analysis of the techniques user-defined parameters including those required and optional
3. Analytical analysis of the proposed technique including a model to describe its empirical behaviour
4. Extension of the proposed technique in terms of its implementation details with the intent of improving either or both classification accuracy and efficiency. This may also include an investigation into making one or more of the existing user parameters automatically adaptable
5. An investigation into the techniques applicability to other engineering problem domains already visited by CLONALG, such as static and dynamic function optimisation, combinatorial optimisation such as TSP and pattern recognition problems such as binary character recognition

It was mentioned in [11], that a technique called and AIS aiNET is an extension of CLONALG though integrated into the network theory of immunology. Given this insight, it may be a useful exercise to investigate the nature of the technique and its applicability and adaptability to classification problem domains. Given the strong link between CLONALG's implementation and evolutionary algorithm, specifically Evolutionary Strategies, it may be a useful exercise to investigate the application of ES to classification domains. Further the a review of Learning Classifier Systems (LCS), and specifically those LCS inspired from the field of parallel niching genetic algorithms such as implicit niching (based on fitness sharing) may provide useful insight into the development of more robust, and efficient immune-inspired classifier systems.

7. Conclusions

The clonal selection theory for acquired immunity has been shown to be a useful metaphor for solutions for complex engineering problems such as pattern recognition, combinatorial optimisation and function optimisation. The work has demonstrated that the powerful Darwinian-like metaphor can be used as the basis of classifier system, the preliminary results of which indicate robustness, insensitivity to parameters and the capability of constructing competitive classification systems.

Given a review of the primary clonal-selection inspired algorithm – CLONALG algorithm and its variant implementations the Clonal Selection Classifier System (CSCA) technique was conceived, designed, specified and preliminary tested. Useful user-feedback algorithm statistics were defined to aid in the further development of the technique and in analysing both CSCA and potentially other clonal selection based techniques with the intent of better understanding and improving the techniques. It is expected that through the rigorous testing and analysis of the proposed CSCA classifier system, that a practical and useful addition to the field of immune-inspired classification will be realised.

8. Appendix – WEKA Algorithm Implementation

WEKA is a machine learning workbench [12] written in the Java programming language. It provides a large number of common classification, clustering, attribute selection algorithms as well as visualisation tools, algorithm test schemes and data filtering tools. WEKA provides a number of interfaces for making use of the tools and algorithms provided such as a command line interface, a data exploration interface, an algorithm test and comparison interface, a workflow interface, and finally a programmer level application programming interface for integrating WEKA functionalities into standalone allocations. WEKA has been made open source, allowing academics and industry to extend the platform by adding algorithm and tool plug-ins for the platform.

As apart of this work investigating the CLONALG algorithm, a naïve classifier implementation of CLONALG was prepared and tested for WEKA. As far as the author knows, there does not exist an implementation of the CLONALG algorithm for professional and academic level application that is both documented and open source. Also implemented for the WEKA platform was the proposed Clonal Selection Classifier System (CSCA) algorithm. It should be noted that the version of CSCA provided is preliminary and should be considered experimental until further testing and analysis is performed to refine the technique.

The provided implementation of CLONALG and CSCA for WEKA offers the following advantages:

1. Standardised programmatic and user interface
2. Java implementation (platform independent)
3. Open-source, taking advantage of refinements and improvements from contributors
4. Allows CLONALG and CSCA to be made available to the wider user base of WEKA users (both academic and industrial)
5. Provides a convenient, consistent, proven and tested platform for algorithm benchmarking, testing and comparison
6. Take advantage of the wide array of classifier-specific performance measures, visualisation tools, standard datasets and data filtering and preparation tools
7. Prepared models (including classification models), as well as algorithm configurations can be saved externally, reused and distributed

Both algorithm implementations are located in the package “weka.classifiers.immune” and are accessed as such in the WEKA user interfaces from the algorithm selection drop-down.

8.1 Clonal Selection Algorithm (CLONALG)

Although the focus of this work was to develop a new clonal selection classifier algorithm, an implementation of CLONALG was prepared as defined in [4,5]. The naïve implementation has a separate memory and remaining antibody pool as defined in the CLONALG specification and allows random number of antibodies to be generated and

inserted each generation. Affinity was implemented the same as specified for CSCA, and classification was performed by always selecting the best matching antibody. All selection, deletion and antibody management was performed using affinity of the antibody to a single antigen, thus antigens were exposed to the system one at a time.

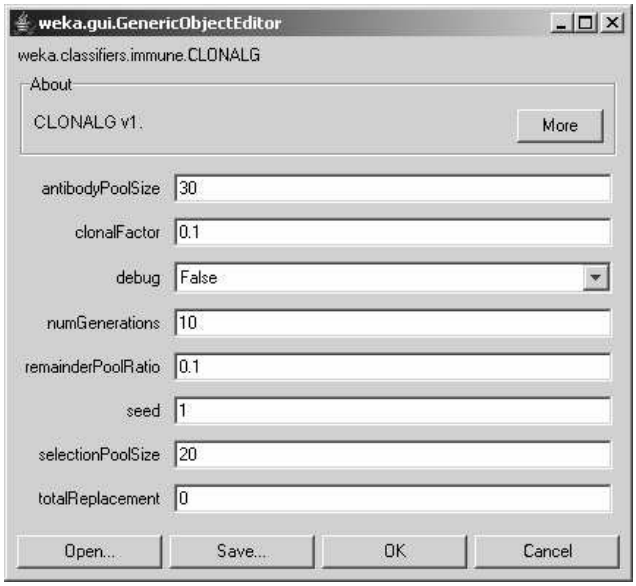


Figure 10 - Shows the CLONALG configuration panel from the WEKA interface

The naïve CLONALG implementation has the following user-defined parameters

1. **antibody pool size (*N*)** – the total number of antibodies in the system, this then divided into a memory pool and a remainder pool
2. **clonal factor (*β*)** – The factor used to scale the number of clones created for each selected antibody
3. **number of generations (*G*)** – the total number of generations to run for. A single generation consists of an iteration through all antigens
4. **remainder pool ratio** – the ratio of the total antibodies (*N*) to allocate to the remainder antibody pool (used for new antibody insertions)
5. **random number generator seed** – The seed for the random number generator
6. **section pool size (*n*)** – the total number of antibodies to select from the entire antibody pool for each antigen exposure
7. **total replacements (*d*)** – the total number of new random antibodies to insert into the remainder pool each antigen exposure

8.2 Clonal Selection Classification System (CSCA)

The CSCA implementation matches the specification for both the procedure and parameters listed in section 5. All the statistics listed in that section are also implemented, and can be retrieved from the algorithm by setting the debug parameter to true (true by default). It is important to note that although the implementation (code) has been tested, the technique has not undergone rigorous testing, and thus should be considered experimental.

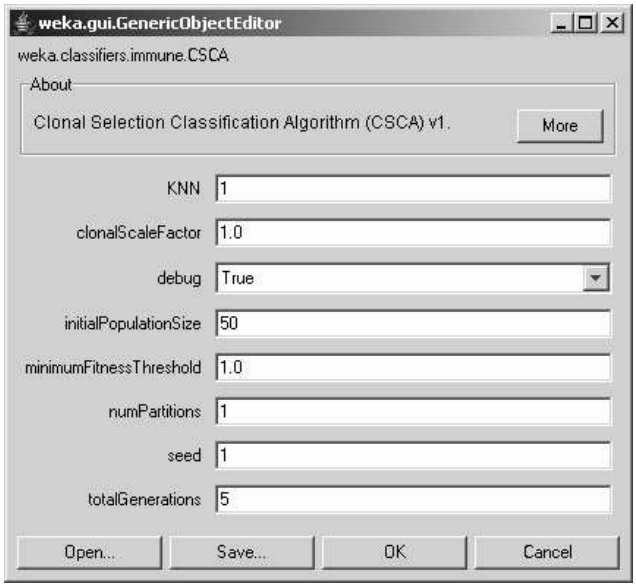


Figure 11 - Shows the CSCA configuration panel from the WEKA interface

8.3 Algorithm Usage

As mentioned, the CSCA and CLONALG implementations can be used directly from the WEKA Explorer WEKA Work Flow, and WEKA Experimenter interfaces. For the algorithm implementations to be recognised by WEKA, the CSCAWeka.jar file must be in the Java class path. The implementation was prepared with Java 5.0 (1.5), and thus the installed Java Runtime Environment (JRE) must be also be this version. Finally, the version of WEKA that the algorithm was prepared for and tested with is 3.4.3.

The following provides examples of using the two algorithm implementations from the command line on the Iris Plants dataset with 10-fold cross-validation.

```
java –cp weka.jar;CSCAWeka.jar
weka.classifiers.immune.CLONALG -B 0.1 -N 30 -n 20 -D 0 -G
10 -S 1 -R 0.1 -t data/iris.arff

java –cp weka.jar;CSCAWeka.jar weka.classifiers.immune.CSCA
-D -S 50 -G 5 -r 1 -a 1.0 -E 1.0 -k 1 -p 1 -t data/iris.arff
```

Figure 12 - Shows examples of executing the two WEKA implementations from the command line

The following code sample provides an example application of using the CSCA algorithm in standalone mode. The program loads the Iris Plants dataset and performs a 10-fold cross-validation test.

```
public class SimpleCSCAUsage
{
     public static void main(String[] args)
```

```
     {
          try
          {
            // prepare dataset
            Instances dataset = new Instances(
              new FileReader("data/iris.arff"));
            dataset.setClassIndex(dataset.numAttributes()-
1);
            CSCA algorithm = new CSCA();
            // evaulate
            Evaluation evaluation = new Evaluation(dataset);
            evaluation.crossValidateModel(algorithm,
              dataset, 10, new Random(1));
            // print algorithm details
            System.out.println(algorithm.toString());
            // print stats
            System.out.println(
              evaluation.toSummaryString());
          }
          catch(Exception e)
          {
               e.printStackTrace();
          }
     }
}
```

Figure 13 - Shows Java code implementation of an application using CSCA to classify the Iris plants dataset

9. Bibliography

[1] Eli Benhamini, Richard Coico, and Geoffrey Sunshine. *Immunology - A Short Course* , USA: Wiley-Liss, Inc., 2000.

[2] William E. Paul. *Immunology - Recognition and Response : Readings from Scientific America magazine / edited by William E. Paul.*, USA: Scientific American Inc., W. H. Freeman and Company, 1991.

[3] John W. Kimball. *Introduction to Immunology*, New York, USA: Macmillan Publishing Co., 1983.

[4] Leandro N. de Castro and Fernando J. Von Zuben, "The Clonal Selection Algorithm with Engineering Applications," *GECCO 2000, Workshop on Artificial Immune Systems and Their Applications,* Las Vegas, USA, pp. 36-37, 2000.

[5] Leandro N. de Castro and Fernando J. Von Zuben, Learning and Optimization Using the Clonal Selection Principle *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, vol. 6, pp. 239-251, 2002.

[6] Andrew Watkins, Jon Timmis, and Lois Boggess, Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm *Genetic Programming and Evolvable Machines*, vol. 5, pp. 291-317, Sep, 2004.

[7] Jennifer White and Simon M. Garrett, "Improved Pattern Recognition with Artificial Clonal Selection," *ICARIS-2003,* Edinburgh, pp. 181-193, 2003.

[8] Andrew Watkins, Xintong Bi, and Amit Phadke, "Parallelizing an Immune-Inspired Algorithm for Efficient Pattern Recognition," *Intelligent Engineering Systems through Artificial Neural Networks: Smart Engineering System Design: Neural Networks,* pp. 225-230, 2003.

[9] Joanne H. Walker and Simon M. Garrett, "Dynamic Function Optimisation: Comparing the Performance of Clonal Selection and Evolutionary Strategies," *ICARIS-2003,* Edinburgh, pp. 273-284, 2003.

[10] Hunter Rudolph. Evolution strategies. In: *Evolutionary Computation 1 - Basic Algorithms and Operations*, eds. Thomas Back, David B Fogel, and Zbigniew Michalwicz. UK: Institute of Physics Publishing, 2000.pp. 81-88.

[11] Simon M. Garrett, "Parameter-free, Adaptive Clonal Selection," *Congress on Evolutionary Computing,* Portland Oregon, USA, 2004.

[12] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*, San Francisco: Morgan Kaufmann, 2000.

[13] C. L. Blake and C. J. Merz. UCI Repository of Machine Learning Databases. [Online] http://www.ics.uci.edu/~mlearn/MLRepository.html . 98. University of California, Irvine, Dept. of Information and Computer Sciences.

[14] Wlodzislaw Duch. Logical rules extracted from data. [Online] http://www.phys.uni.torun.pl/kmk/projects/rules.html . 2002. Computational Intelligence Laboratory, Department of Informatics, Nicholaus Copernicus University, Torun, Poland.

[15] Wlodzislaw Duch. Datasets used for classification: comparison of results. [Online] http://www.phys.uni.torun.pl/kmk/projects/datasets.html . 2002. Computational Intelligence Laboratory, Department of Informatics, Nicholaus Copernicus University, Torun, Poland.