Immunology as a Metaphor for Computational Information Processing: Fact or Fiction ?

Emma Hart

Doctor of Philosophy

Artificial Intelligence Applications Institute
Division of Informatics
University of Edinburgh

2002

Abstract

The biological immune system exhibits powerful information processing capabilities, and therefore is of great interest to the computer scientist. A rapidly expanding research area has attempted to model many of the features inherent in the natural immune system in order to solve complex computational problems. This thesis examines the metaphor in detail, in an effort to understand and capitalise on those features of the metaphor which distinguish it from other existing methodologies. Two problem domains are considered — those of scheduling and data-clustering. It is argued that these domains exhibit similar characteristics to the environment in which the biological immune system operates and therefore that they are suitable candidates for application of the metaphor. For each problem domain, two distinct models are developed, incorporating a variety of immunological principles. The models are tested on a number of artifical benchmark datasets. The success of the models on the problems considered confirms the utility of the metaphor.

Acknowledgements

My grateful thanks to Professor Peter Ross for his invaluable help and guidance throughout the duration of this project. Also to my husband Jim, firstly for proof-reading this thesis, but most of all for his encouragement and support, and belief that it was possible to finish this work and simultaneously look after our two young daughters!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Emma Hart*)

To Lucy and Holly

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The study of biological systems has long proved inspirational to the computer scientist as a means of solving complex computational problems, with many attempts to mimic the mechanisms inherent in the natural world. For example, early neural network pioneers attempted to model the circuitry and processing thought to be found in the brain; the field of evolutionary algorithms was inspired by Darwininan studies of natural evolution; ant colony optimization algorithms are modelled on the behaviours exhibited by real ants, and more recently, the concept of DNA computing has arisen, inspired by the processes that govern life itself. The driving force behind such research is two-fold: the use of biologically inspired metaphors can result in new computer technologies and novel methods of problem solving, and conversely, computing can provide new tools and techniques for exploring biological concepts from an alternative prospective.

The field of *artificial immune systems* (AIS) is also inspired by a biological metaphor — in 1986 the theoretical immunologist, J.D. Farmer, first suggested a possible relationship between biological immunology and computing in a paper which compared natural immune systems, adaptation and machine learning ([Farmer et al., 1986]). Since then, the field has expanded rapidly, with numerous papers published by computer scientists applying AIS to a diverse set of topics ranging from computer security [Forrest et al., 1994] to behaviour arbitration for autonomous mobile robots [Ishiguro et al., 1996]. A dip into the biological journals reveals a similar number of computational models of immunological phenomena, for example [Weinand, 1990, Perelson, 1989, Celada and Seiden, 1992]. In light of the increasing

amount of research in this area, it thus seems pertinent to examine the immune system metaphor in relation to information processing in more detail, and to ask which are the features of the natural immune system that really distinguish it from other biological metaphors, and to attempt to categorise the types of problem area in which this particular metaphor might provide an advantage over others.

1.1 Immunology

There are four main causes of death to the human being — injury, infection, degenerative disease and cancer. Of these, only the former two regularly kill their victims before they reach child bearing age, and as such are a potential source of lost genes. The immune system is an example of a mechanism which may help to ensure the survival of those genes, and has evolved over time in order to protect us from infectious organisms existing in the environment. Thanks to the immune system, infections in a normal individual caused by microbes such as viruses, bacteria, fungi and parasites are generally short lived and leave little permanent damage. The immune response broadly falls into two categories — the *innate* or non-specific response, and the *adaptive* or specific response. The innate response is provided by a number of non-specific chemicals such as lysozyme which destroys the outer surface of many bacteria, non-specific chemical effectors such as macrophages and simple barrier mechanism such as the skin. On the other hand, the adaptive response is highly specific for particular pathogens (antigens), and furthermore, it improves with each subsequent exposure to the pathogen. It can therefore can be said to 'remember' specific pathogens. It is with this adaptive aspect of the immune system that artificial models are generally concerned. The adaptive response consists of two major phases — a recognition phase followed by a reaction to eliminate the pathogens, and is achieved via a class of immune cells collectively known as *lymphocytes*. Recognition is generally accepted to require the immune system to be able to distinguish between the body's own cells (*self*) and foreign pathogens, (*non-self*), though recently some immunologists have controversially rejected this theory and proposed that in fact the job of the immune system is only to distinguish *dangerous* non-self from self ([Matzinger, 1994a, Matzinger, 1994b]. As far as computer

scientists are concerned however, the task essentially remains one of recognition followed by an action such as elimination, and it is the mechanisms by which the natural immune system achieves these aims that make the system so attractive to the topic of information processing.

1.2 Some Key Concepts and their Relevance to Information Processing

The immune system can be considered to be a remarkably efficient and powerful information processing system which operates in a highly parallel and distributed manner. It operates in a dynamic and unpredictable environment in which it is necessary to react to changes in a timely manner — this is achieved partly through imprecise but efficient recognition mechanisms and by utilising memories of past experiences to provide useful pointers to the correct course of action. It contains several features that make it appealing from a computational point of view. These are summarised below. The list attempts to correlate features of immune system with the well-known terminology of information processing. The information is adapted from that given in [Dasgupta, 1998]:

- *Recognition:* The immune system can recognise and classify different patterns and generate selective responses. In the natural immune system, recognition is achieved via inter-cellular binding, the strength of which is determined by molecular shape and electrostatic charge. One view is that during the recognition process, the immune system is solving the problem of self-nonself discrimination.

- *Feature extraction:* Features are extracted from pathogens by antigen presenting cells or APCs which extract features from them and present them on their surface. This serves two purposes, that of a filter and a lens. The filter removes noise and the lens focuses attention.

- *Diversity:* The immune system can utilise a combinatoric process to generate a diverse set of pathogen recognising molecules, and ensures that at least some

lymphocytes can bind to any antigen, whether known or unknown.

- *Learning:* The immune system learns by experience the structure of specific antigens, following the first exposure (primary response) of the system to a new antigen. The main mechanism for learning is via altering the concentrations of lymphocytes during the primary response phase.

- *Memory:* It has been shown that when the immune system has been activated, a few lymphocytes become special 'memory cells' which are then content-addressable. The longevity of these cells is dynamic and requires continued stimulation from residual antigens. A balance is achieved between economy and performance by maintaining a minimal but sufficient memory of the past.

- *Distributed detection:* The immune system is inherently distributed throughout the body — lymphocytes constantly circulate throughout the blood, lymphoid organs and tissue spaces.

- *Self-regulation:* There is no central organ coordinating the immune response and therefore the mechanisms are self-regulatory, although not necessarily stable in the sense of converging to a time-independent state.

- *Threshold Mechanism:* An immune response and the subsequent proliferation of immune cells only takes place above a certain matching threshold, related to the strength of chemical binding.

- *Co-Stimulation:* Activation of immune cells is regulated through co-stimulation in which 'helper' T-Cells deliver a second signal, to ensure tolerance and to distinguish between harmless and dangerous invaders.

- *Dynamic protection:* The processes governing generation of high-affinity immune cells dynamically balance exploration vs exploitation in adaptive immunity. This dynamic protection increases the cover provided by the immune system over time.

- *Probabilistic detection:* Detection of antigens is approximate, therefore a lymphocyte can bind with several different kinds of structurally related antigen.

Thus, the immune system contains a number of general mechanisms which potentially can be copied or adapted in computer systems. From the perspective of information processing, it is unnecessary to attempt to replicate *all* of these aspects of the natural immune system in a computer model, rather they should be used as general guidelines in designing a system. Indeed, as becomes clear in the literature review of chapter 2, in practice most AIS applications only implement some modified subset of these features. Perhaps more importantly, it should be noted that several of these features are apparent in other biologically inspired systems — the IS has been compared to artificial neural networks ([Dasgupta, 1997], to sparse distributed memories (SDM) [Smith et al., 1999], to classifier systems [Farmer et al., 1986] and to case-based reasoning systems [Hunt et al., 1995]. Therefore, a question that deserves more attention is to what end does the immunological metaphor provide analogies that *cannot* be provided by another less seductive labelling. One of the aims of this thesis is to attempt to isolate the unique features of the immune system that seem most relevant and identify the types of problem areas to which they could profitably be applied.

1.3 Overview of Application Areas

The above discussion suggests that potential application areas for the application of the immune system metaphor are those in which we are seeking robust and 'good enough' solutions to problems occurring in dynamic environments that allow a system to continue functioning satisfactorily. These features are characteristic of a number of real-world problem domains. However in this thesis two particular areas are chosen as being particularly suitable; scheduling and data-clustering. As will be seen in the next section when the analogy is made explicit, the problems faced by the immune system of recognising and eliminating harmful invaders on a relatively short timescale are very similar to those faced in the two identified domains.

1.3.1 Scheduling

Consider a typical real-world manufacturing scenario in which assembling a finished product for delivery to a customer requires the manufacturing of several individual

parts or subcomponents of the product before they can be assembled into the final article. The rate of production of the products and hence ultimately the cost of producing them is controlled by a *schedule* — however, creation of a suitable schedule is well-known to be a highly complex problem. Many factors must be taken into consideration when producing a schedule, for example the costs associated with storing the raw materials required to manufacture the products, the common need to produce expensive products on a 'just-in-time' basis, and the set-up times associated with using machines. Even when an attempt is made to take these factors are taken into account, the fact cannot be ignored that a factory is by definition operating in a dynamic and unpredictable environment:- machines break down, employees get sick, materials arrive late, and customer demands change rapidly. Some of these events occur frequently, and can be more or less predicted to some extent (for example, materials from a certain supplier may often arrive late), whereas others occur on a much more ad-hoc basis and cannot be foreseen. An 'ideal' schedule therefore, if there is such a thing, is not necessarily one which optimises some measurable criterion such as make-span or maximum tardiness, but one which has some built-in flexibility that can absorb some unpredictable event without disrupting the planned schedule. At the same time, the schedule should still deliver some acceptable level of quality when measured against some pre-determined criteria.

Thus, it can be seen that the task of producing robust schedules has a direct analogy with the task faced by the immune system . Both operate in a dynamic and unpredictable environment — the immune system must mount an efficient and immediate antibody response against invaders if it is to survive — similarly, in order to minimise costs, a useful scheduling system should be able to mount a response to environmental changes by rapidly altering schedules so that minimum disruption is caused. The antibodies produced by the immune system do not have to perfectly match the invading pathogens in order to eliminate them, similarly the new schedules produced by the scheduling system to not have to be optimal, just 'good enough' for the scheduling to continue with the least interruption. Futhermore, both systems can utilise a memory of past events in order to produce an efficent response, but are also required to be capable of responding to entirely new situations. It therefore seems plausible that some or all

of the characteristics of the immune system may be adapted to implement a scheduling system.

1.3.2 Data Clustering

Modern technology makes it incredibly straightforward for companies to gather vast amounts of data concerning individuals and their habits on a daily basis, for example through the use of credit cards or supermarket loyalty cards. Interpreting such huge quantities of data, and identifying clusters and trends within it is a mammoth task, especially as the data may be rapidly changing. Data-clustering can be defined as *"the unsupervised classification of patterns (observations, data items or feature vectors) into groups (clusters)"* [Jain et al., 1999], and is performed in the hope that implicit previously unknown and potentially useful knowledge can be extracted from the data. It is a large field in its own right, and there are many documented approaches. The reader is referred to [Jain et al., 1999] for a recent and detailed survey.

However, the immune metaphor may provide a novel and alternative approach. Both the immune system and a data-clustering system have to operate in very large input spaces. In the immune system, a lymphocyte recognises a *set* of antigens, due to its imprecise matching characteristics; that set can be considered to be equivalent to a cluster within a database. The lymphocyte that recognises all the items in a cluster thus provides a concise description of the cluster itself. The number of lymphocytes present and the specificity of the recognition process provides a mechanism for controlling the number of clusters present, and hence provides a method of controlling how specifically (or generally) the clusters are described. The fact that recognition is imprecise is important — data in a database is likely to contain much noise and redundant information, therefore some kind of imprecise recognition mechanism will be essential.

The natural immune system can react to unseen pathogens either by producing new lymphocytes using its inbuilt diversity generating mechanisms or by adapting existing lymphocytes via mutation mechanisms. Similarly, when new data arrives in the database, the centres and sizes of the clusters may need to move and adapt in order to recognise the new data. New cluster centres may be created and old ones may

disappear over the course of time, the key point being that the system can respond dynamically to the state of the database at a given moment in time.

The natural immune system is very efficient at recognising the sudden appearance of harmful invaders; a data-clustering system should be able to recognise the appearance of anomalous data in the database. This feature would automatically result from an immune based model — data-items belonging to existing clusters would be recognised by existing lymphocytes in the system's memory, whereas data belonging to new clusters would trigger creation of an entirely new lymphocyte. This event could trigger a warning to an external observer of the system, signifying that the new item is non-representative of the general patterns. Imagine for example attempting to cluster data collected by a credit-card company relating to card usage. The company is interested in clustering the data to identify patterns in card usage, but would also like to detect fraudulent card-usage. If a newly presented data-item does not belong to an already established cluster, it could identify an attempt at fraudulent usage of the card, which further human examination could verify.

Finally, the distributed nature of the immune system architecture is attractive, given the fact that very large datasets are also likely to be distributed. This dissertation presents two new immune system based models for tackling non-stationary data-clustering problems, that attempt to take advantage of the immune metaphors.

1.4 Aims and Contributions of Thesis

The main contributions of this dissertation are as follows:

1. Development of two AIS models for performing job-shop scheduling, both based on the use of evolutionary algorithms.

2. Analysis of the models, and empirical testing and comparison of them on benchmark job-shop scheduling data.

3. Development of two models for performing clustering in non-stationary databases based on combining immune system metaphors with that of another class of associative memories, SDM.

4. Analysis and testing of both models on an artificially generated test-bed of non-stationary data.

1.5 Guide to Remainder of Thesis

Chapter 2 introduces some basic immunology for computer scientists. This is followed by a review of a number of very different models of artificial immune systems, which identifies the features of the natural immune system each model contains, and discusses the types of application to which each has been applied. Chapter 3 contains a detailed description of the scheduling domain and presents two AIS models for solving job-shop scheduling problems, including detailed experimental results. In Chapter 4, the data-clustering domain is described, and an artificial test-bed for experimenting with AIS models is introduced. Chapter 5 presents an AIS which is evolved using a co-evolutionary genetic algorithm, and describes the results of extensive experimentation. An improved AIS for performing data-clustering that is self-organising is then presented in Chapter 6, with new results. The dissertation is concluded in Chapter 7.

Chapter 2

Background

This chapter begins with a brief introduction to immunology, necessary to set the scene for the remainder of the thesis. It is of course a vast topic, and only the most relevant features are covered here. For a more detailed overview, the interested reader is referred to an introductory immunology text such as [Roitt et al., 1988]. This introduction is followed by a review and comparison of existing AIS implementations, and an overview of existing literature in the application of immune systems to the chosen domains of scheduling and data-clustering. Finally, there is a discussion of the relationship of the immune system to a class of associative memories known as Sparse Distributed Memories.

2.1 Basic Immunology

This section presents some basic immunological concepts which are central to the adaptive immune response. As already stated, it is with this aspect of the immune system that most artificial systems are concerned. Key to all adaptive responses is a class of cells known as lymphocytes which specifically recognise individual pathogens, regardless of the location of those pathogens, whether in blood, tissue fluids or actually inside host cells. Lymphocytes fall into two categories, T-Cells (*T*hymus dependent), and B-Cells, (*B*one marrow dependent). The function of the B-Cell is to attack extracellular pathogens by releasing antibodies, i.e. specific molecules which recognise and

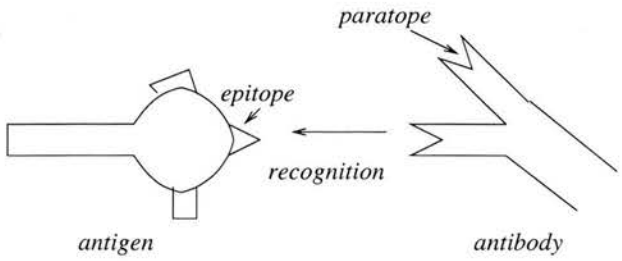

Figure 2.1: 'Lock and Key' recognition between antigen and antibody

bind to target antigens. Antigens can be either a molecule on the surface of a pathogen, or a toxin produced by the pathogen. The antibodies have a distinct molecular structure, that of a flexible Y-shape, and recognise the shape of particular antigen via a mechanism often likened to a *lock and key*, as shown in figure 2.1. The portion of the antigen that is recognised by the antibody (and therefore acts as the lock) is known as the *epitope* (antigen determinant), and the portion of the antibody analogous to the key that recognises the antigen determinant is known as the *paratope*.

T-Cells have a wider range of functions. One group of T-Cells interacts with the B-Cells to help them divide, differentiate and make antibodies. Another group interacts with phagocytic cells (which bind to micro-organisms and internalise them) to help them destroy intra-cellular pathogen. These two groups are known as helper T-cells. The third kind recognises cells infected by viruses and destroys them. In general, most implementations of artificial immune systems have concentrated on mimicking the functionality of B-Cells and ignored the role of T-Cells, though some aspects of helper T-Cells are modelled in some systems, for example [Carter, 2000].

There is much immunological evidence to verify the existence of the basic cells involved in the immune response, however opinion as to the process by which these cells are able to mount a response falls into two distinct camps. One camp favours a process known as *clonal selection*, the other argues for the existence of an immune network. As both approaches have potential significance for artificial models of the immune system, they are now presented.

2.1.1 The Network Hypothesis

Studies have shown that each antibody has a specific antigen determinant known as the *idiotope* — this gives rise to a possibility first articulated by Jerne in [Jerne, 1973] that antibodies can recognise other antibodies as well as antigens, resulting in a large, self-regulating and mutually reinforcing network of antibodies.

This is shown in figure 2.2. In this diagram, the idiotope of B-Cell1, *ID1* stimulates B-Cell2, and the two become connected via the paratope of B-Cell2, $P2$. Thus, $ID1$ is acting as an antigen from the viewpoint of B-Cell2, and this causes B-Cell2 to suppress the antibodies produced by B-Cell1. On the other hand, $ID3$ acts as an antigen from the viewpoint of B-Cell1, and is recognised by BCell1s paratope, $P1$, and thus $ID3$ stimulates B-Cell1 to produce antibodies. Hence, a large chain of suppression and stimulation can be set up between B-Cells, resulting in a self-organising and self-regulatory network. Importantly, the network is not fixed, but varies continuously according to the dynamical changes in the environment. This is known as the *meta-dynamics* of the system [Varela and Coutinho, 1988], and is realised by incorporating newly generated cells into the network and removing useless ones. The new cells are generated when cells in the existing network are stimulated and proliferate, resulting in some mutant species, and also owing to gene-recombination in the bone marrow. Despite these dynamic perturbations, an underlying core network of B-Cells is thought to be maintained by the immune system, representative of the antigens to which it has been exposed. The network remains stable due to the suppression mechanisms which prevent the over-stimulation of B-Cells.

2.1.2 Clonal Selection

The clonal selection theory considers that each lymphocyte, whether B-Cell or T-Cell, is capable of recognising essentially one kind of antigen. When an infectious agent is encountered, a few of the many circulating B-Cells recognise it. Those cells are then induced to proliferate rapidly until within a few days there are sufficient number of them to mount an adequate response. This process by which an antigen selects for and generates the specific clones of its own antigen-binding cells is known

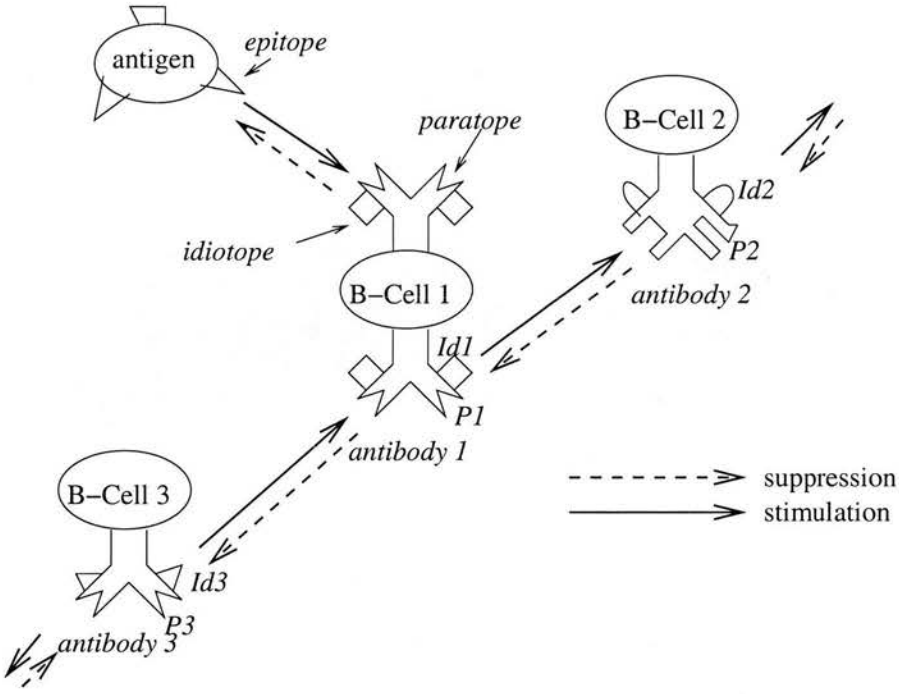

Figure 2.2: Jerne's Idiotypic Network Hypothesis

as clonal selection and is illustrated in figure 2.3, which is adapted from that given in [Roitt et al., 1988].

Those lymphocytes that are stimulated by binding to specific antigen begin to undergo cell-division by expressing new receptors which signal proliferation. Several cycles of division occur, before some of the proliferating B-Cells eventually mature into *plasma* cells which are capable of producing antibodies specific to the antigen. Others mature into *memory* cells, which retain the immunological memory of the stimulating antigen and are then available for re-stimulation should re-infection with the antigen occur at a later date. Thus, the memory cells confer long lasting immunity on the system. During proliferation, some of the daughter cells may undergo somatic mutation which can increase the specificity of the antibody for the antigen — this effect is discussed in more detail later in this chapter in section 2.2.2.1. Clearly this overview represents an extreme simplification of the actual processes that occur during proliferation of the antibodies, in particular it omits the role of helper T-Cells which assist in the proliferation of B-Cells. Nevertheless, the detail provided is sufficient to al-

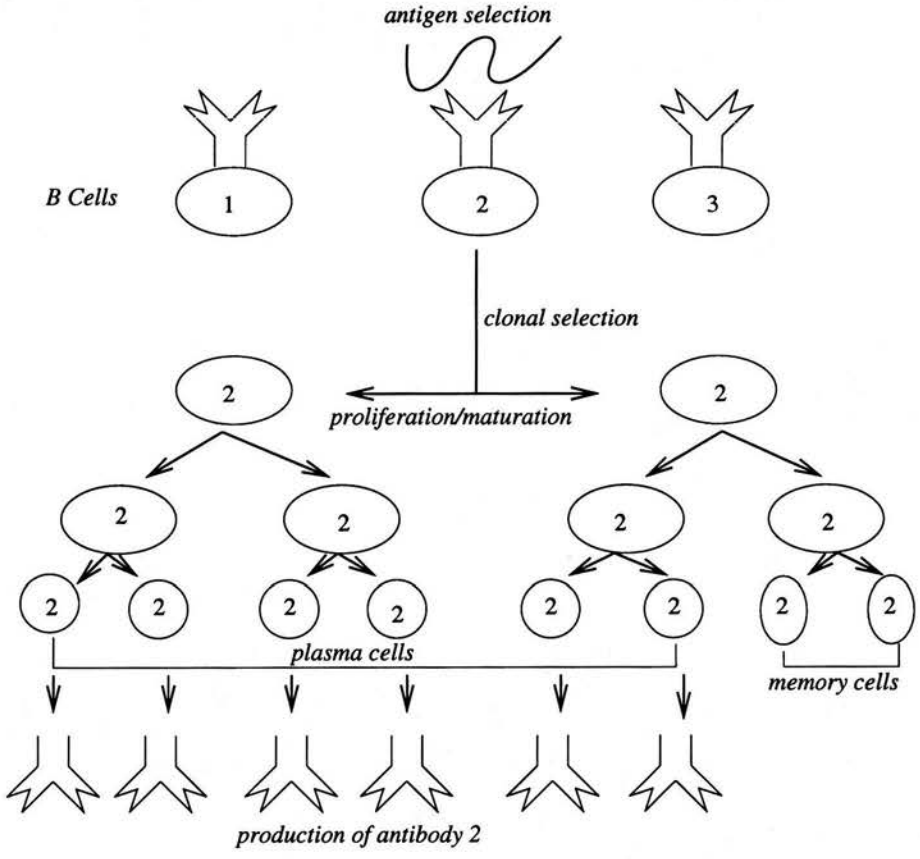

Figure 2.3: B-Cell Clonal Selection

low the main processes apparent in the biological immune system to be captured and implemented in an artificial system.

It is unnecessary for computer scientists to be drawn into a debate about which of the two hypotheses presented as alternatives for the mechanism by which the real immune system operates is correct. Simply, it is sufficient to note that *both* hypotheses contain important properties that have potential analogies as far as information processing is concerned, and that therefore any combination of these ideas may be modelled in a computational immune system when applying the metaphor to fields such as scheduling and data-clustering.

2.2 Artificial Immune Systems

The seminal work that kick-started the field of Artificial Immune Systems was a paper by Farmer and Perelson [Farmer et al., 1986] in 1986. This paper introduced a dynamic model of the immune system based on Jerne's network hypothesis that was simple enough to simulate on a computer. An antibody is represented as pair of binary strings (p, e) signifying the paratope and epitope. A simplifying assumption is made that each antibody consists of exactly one paratope and one epitope although in reality this is not the case. Reaction between antibodies (and between antibodies and antigens) is simulated via complementary matching of strings. The method in which matching is implemented attempts to model several features of the biological system. Firstly, exact matches are not required for reaction to take place and secondly, strings are allowed to match in more than one alignment, with the strength of the match proportional to the sum of all possible matches. This is an attempt to model the fact that molecules can interact in more than one way, and if so, react more strongly as they spend more time together than those molecules that can only interact in one alignment. A *threshold* is introduced, as in the natural IS, below which reaction is not considered to take place.

From a microscopic point of view, when two antibodies interact, the antibody with the paratope reproduces some fixed number of times, whilst the antibody with the epitope is eliminated with some fixed probability. This is controlled by the degree of complementarity between the paratope and epitope. Simulation of the microscopic

dynamics is avoided in the model by use of differential equations for the concentrations. If there are N antibody types, with concentrations $x_1, x_2, ..., x_N$ then the system is simulated via the following differential equation:

$$\dot{x}_i = c\left[\sum_{j=1}^{N} m_{ji}x_i x_j - k_1 \sum_{j=1}^{N} m_{ij}x_i x_j + \sum_{j=1}^{N} m_{ji}x_i y_j\right] - k_2 x_i \tag{2.1}$$

The first term represents the stimulation of the paratope of antibody i by the epitope of antibody j. The second term represents the suppression of antibodies of type i when their epitopes are recognised by the paratopes of type j. Both terms assume that the probability of a collision between an antibody of type i and j is proportional to the product of the concentrations of these antibodies $x_i x_j$. The third term captures the fact the system is driven by the presence of antigens, of concentration y_i. In these three terms, the match specificities m_{ij} take into account what reactions occur and how strongly. Finally, the last term models the tendency of antibodies to die in the absence of any interactions. The parameter c is a rate constant which depends on the number of collisions per unit time, and the rate of antibody production stimulated by a collision. k_1 represents a possible inequality between stimulation and suppression and k_2 is a further rate constant which can be varied. An essential element of the model is that the list of antibody and antigen types is dynamic — new antibodies are generated by applying crossover and mutation operators to the paratopes and epitopes of existing antibodies, and antigens are generated either randomly or by design.

Farmer also postulates that the idiotypic network formed in this model provides a mechanism for allowing antigen to be remembered for long periods of time, bearing in mind that in some cases, antigens in the biological immune system can be remembered over time periods comparable to the lifespan of the organism. Consider figure 2.4 adapted from [Farmer et al., 1986]. Paratope p_i recognises epitope e_{i-1} for $i = 1, 2, ..., n$. If by chance p_1 recognises e_n in addition to e_0 then a cycle is formed, and e_n must resemble e_0. If the antigen is eliminated, then the existence of the cycle maintains the concentration of those antibodies that recognised the antigen and thus provides a memory of the antigen.

Thus, this model incorporates many of the features of the natural immune system. The preliminary paper by Farmer does not report any application of the model to a

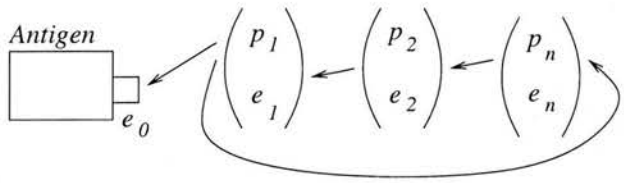

Figure 2.4: Formation of a cycle allowing antigen with epitope e_0 to be remembered. The arrows denote recognition via the matching algorithm

pattern recognition system, its primary intention being to learn more about the internal operation of the immune system in real systems. However, they note "...generalised versions of the model may be capable of performing artificial intelligence tasks".

Gibert and Routen [Gibert and Routen, 1994] adopted this approach and attempted to apply it to create a content-addressable auto-associative memory. Inputs to their system are black and white pictures of 64x64 pixels which are analogous to antigens. The aim was to present the antigen to the system, initiate a response during which a memory of the antigen would be created, then observe the existence of the memory by initiating a secondary response via injection of the same or similar antigen. However, they report that they were unable to satisfy the simultaneous requirements of remembering patterns whilst maintaining system stability. They suggest two variations of the model. In the first, they attempted to forcibly create recognition loops in the network to enable the maintenance by the network of clones responding to the antigen, and thus provide a memory of the antigen. However, they show subsequently that this proves unstable, in that clones would proliferate continuously and lead to collapse of the system. They modified this system to increase suppression of clones, which resulted in a stable system, in which memory cells were maintained by the network but tended to dissipate slowly and eventually disappear. However, the system responded poorly, in that it did not show good quality output, particularly after a secondary response.

2.2.1 Negative Selection Based Models

A whole class of implementations of artificial immune systems focus on modelling the generally accepted self/non-self discrimination ability of the biological immune

system. The basic principles behind all of these models are as follows (modified from Dasgupta [Dasgupta and Forrest, 1999])

- Define *self* as a multiset S of strings of length l over a finite alphabet, a collection which we wish to process or monitor
- Generate a set R of *detectors*, each of which fails to match any string in S. (A partial matching rule may be applied)
- Monitor S for changes by continually matching the detectors against S. If any detector ever matches, a change or deviation must have occurred.

This basic algorithm has been employed extensively in computer security applications. [Forrest et al., 1994] applied the analogy to computer virus detection, to host-based intrusion detection[Forrest et al., 1997a], and to making computers robust to wide-spread attacks, [Forrest et al., 1997b]. [Hofmeyr and Forrest, 2000] describe a further system for protecting local area networks (LANs) from network-based attacks. The key to each of these applications clearly lies in defining 'self' in each case. For example, in Hofmeyr's work on LAN security, self is defined as a set of datapath triples defining TCP connections logged to the network. These were collected over a period of 50 days, which after filtering out noisy traffic sources such as web-servers, resulted in a set of 1.5 million datapaths.

The negative detection algorithm has also been applied by Dasgupta in [Dasgupta and Forrest, 1996, Dasgupta, 1996] to detecting anomalies in time series data. In this case, the aim is to detect temporal changes in the cutting force patterns obtained from machine tool data, and thus predict when a machine is likely to break. In this case, self is defined by first collecting raw sensory data from machines in normal operation over a moving time window and mapping this real-valued data into a binary form (essentially by normalising each analog value with respect to a defined range and discretising it into bins — each data point is assigned the integer corresponding to the bin within which it falls).

In [Hofmeyr and Forrest, 2000], Hofmeyr describes a general immune framework called ARTIS, based on the principle of negative selection, which embodies many of the characteristics of the biological immune system. In this system, a set of detectors is

maintained at each of n nodes in a distributed system. The detectors in each set detect non-self and are created and maintained as shown in figure 2.5 which is taken directly from [Hofmeyr and Forrest, 2000]. In this diagram, a detector consists of a randomly created bit-string. New detectors remain *immature* during a tolerization period T in which they are exposed to self (or at worst to an environment which consists mainly of self). If any randomly generated detector matches anything during this period it is killed and is replaced by a new randomly generated detector. If it survives T, it becomes *mature* but *naive*, and lives for a further fixed number of time-steps. The number of matches it accumulates is monitored, and if this number exceeds a certain threshold τ, it becomes activated. Once activated, if it receives some co-stimulation from an outside source to confirm that what is matched was truly non-self, then it becomes a memory detector and lives indefinitely, and from then onwards only requires a single match for activation. If it does not receive co-stimulation, it dies.

Thus, according to Hofmeyr, the immunological principles embodied in ARTIS are as follows:

- It is *distributed*; different detector sets can be placed on different nodes
- Having different detectors at different nodes confers *diversity*
- The system is *robust*, as a loss of some detectors on one node does not result in a complete absence of protection
- If the self set is typical of normal behaviour, then policy is implicitly specified.
- As detection is localized and needs no communication, the system is *scalable*
- The system is *adaptable* to changes in normal behaviour owing to its use of tolerization and finite detector lifetimes.

However, several objections can be raised to the use of this model. Firstly, it is necessary to generate a set of detectors which do not recognise any string in self. The time complexity of this is proportional to the number of times a detector must be regenerated until it is valid. [Forrest et al., 1994] show that the number of retries required to

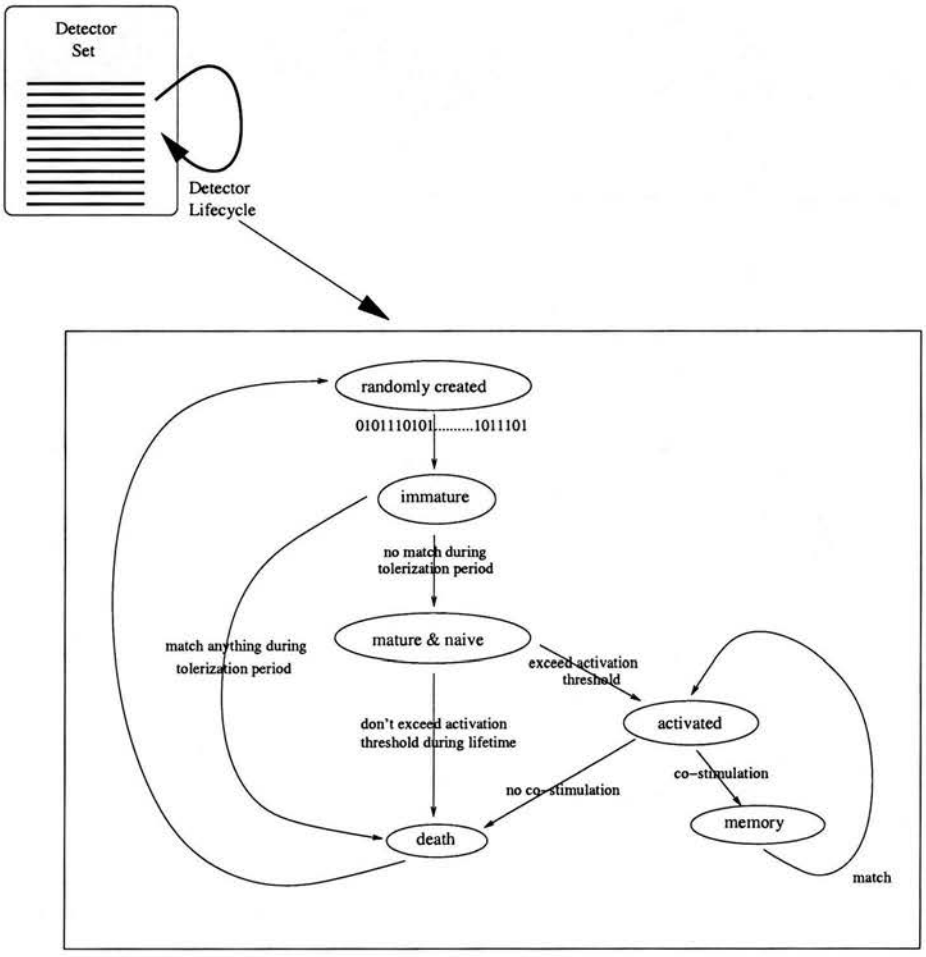

Figure 2.5: The lifecycle of a detector in ARTIS (taken from [Hofmeyr and Forrest, 2000])

generate a valid detector $V(d)$ is a geometric random variable with parameter $P(V(d))$ and so the expected number of trials ρ until success is given by

$$E(\rho) = \frac{1}{(1-\rho_M)^{|S_R|}} \tag{2.2}$$

where S_R is the self set and ρ_M the probability of a match between a candidate detector and a self string. The number of retries is thus exponential with the size of the self set. Helman in [Helman and Forrest, 1994] has proposed an alternative generation algorithm which runs in linear time with the size of self, based on a dynamic programming technique, and [D'haeseleer, 1995] further proposes a greedy algorithm, however these algorithms are not general and only apply to problems in which matching is achieved via a specific matching rule, based on the number of common contiguous matching bits between the detector string and the self-string. Thus, the application of the negative selection model may be limited to domains in which detectors can be generated in a suitable time-frame, and further more to applications in which self can be easily defined.

The second objection and perhaps more fundamental concern is that a system based on negative selection implicitly assumes that the problem domain can be divided into two distinct sets of event, 'normal' and 'abnormal'. In reality, this is not the case, as the categorisation of some events may be ambiguous, depending on circumstances, and hence cannot be correctly classified. ARTIS in particular is built on the assumption that the boundaries between self and non-self can be implicitly inferred by observing the behaviour of the system and assuming that self occurs more frequently than non-self, or that there is some period of time during which self can be collected separately from non-self. Moreover, although the system is adaptable to changes in self, it is unclear that in reality this would be true if those changes occurred on anything but a very slow time-scale.

The final limitation arises from the fact that the system requires the intervention of a human operator to provide the co-stimulation required to convert a mature detector into a memory detector, and hence the system is not completely autonomous, a feature which is of course desirable.

Nevertheless, of all the implementations of artificial immune systems described in the literature, ARTIS perhaps bears most resemblance to a real immune system and has been shown to be capable of performing network intrusion detection. To emphasise the generality of the architecture, [Hofmeyr and Forrest, 2000] also suggest further applications to which the framework could be applied, namely mobile agent security, epidemiological monitoring and detection of fraudulent financial transactions.

2.2.2 Models based on Evolutionary Algorithms

A class of artificial immune systems has emerged from the evolutionary algorithm (EA) community, due to the observation that an EA could act as an excellent tool for evolving sets of antibodies. Three different models that rely on an EA as the underlying engine for producing antibodies are described below.

2.2.2.1 A Library Based Model

The immune systems variable V-region genes contain numerous gene segments, the individual function of which can only be seen when each segment is joined to others to construct one of a large number of possible antibody molecules. It is speculated ([Leder, 1991]) that the human immune system contains seven 'libraries', each containing differing numbers of gene segments and that random selection of a component from each library produces an antibody molecule. As there are many theoretical combinations of these components, an immune system can generate a large number of unique antibodies from a limited genetic source. Hightower [Hightower et al., 1995] and Perelson [Perelson et al., 1996] present an abstract model of such a library bases system. Again using a binary representation as suggested by Farmer in [Farmer et al., 1986] their system uses a haploid chromosome to encode l libraries, each of which contains c gene segments. A gene segment is simply a binary string in each case of length n. An antibody is produced by combining a randomly selected component from each library, as show in figure 2.6, and is thus of length $n \times c$. Thus an immune system containing l libraries, each with c components, can be used to format c^l different antibodies. The complete set of antibodies that can be formed is known as the *potential antibody repertoire*. If the components in each library are

genetically dissimilar, then the scope for producing a set of antibodies that together match a wide range of antigens is increased.

Hightower *et al.* use a genetic algorithm to evolve the composition of the binary immune libraries, i.e. the *genotype* for a task in which the expressed antibodies (the *phenotype*) must recognise a set of binary strings. The fitness of an individual (i.e. the entire genetic library) is determined by its overall ability to recognise antigen molecules. Fitness is determined by generating a set of expressed antibodies from an individual, and testing how well that set recognises a set of antigens. Each *antigen* receives an *antigen score* which is the maximum of all the match-scores computed between the antigen and the expressed antibodies. The overall fitness of the individual is then found by combining the antigen scores, and averaging them. An alternative scheme is to assign the fitness of the individual equal to the *lowest* antigen score, with the rationale that an individuals fitness is effectively limited by that of the antigen it is least equipped to recognise. Despite the fact the fitness pertains only to *phenotypic* information, Hightower *et al* show that a GA is able to effectively organise the structure of the antibody libraries in order to perform this task.

Perelson [Perelson et al., 1996] uses this immunological model combined with a GA to investigate actual biological phenomena such as clonal selection and the Baldwin effect. The latter effect was first observed by Baldwin [Baldwin, 1896] over 100 years ago and is the notion that useful characteristics can be passed down to a future generation without genetic propagation. The learning occurs by a process known as *somatic mutation* in which stimulated antibodies produce daughter cells, in which one or more genes become mutated. The daughter cells thus have varying abilities to recognise a single antigen. Certain key mutations can lead to a significantly increased recognition ability, however, these key mutations are not written back to the genome libraries and hence cannot directly be passed onto future offspring. [Perelson et al., 1996] find that evolution of the genetic libraries can be accelerated by incorporating this type of learning into the model.

To summarise the features provided by this model, its strengths lie in the ability to use a straightforward genetic algorithm acting on a binary representation to evolve a small set of genetic libraries capable of providing a wide range of diversity. The most

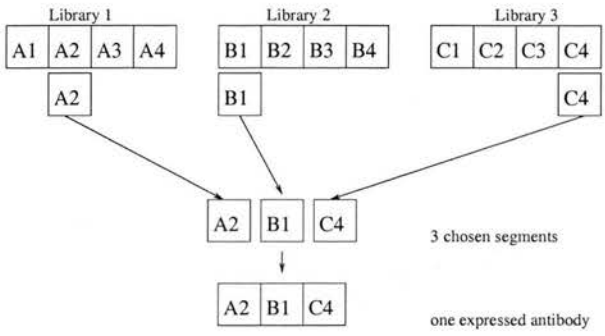

Figure 2.6: Expressing an antibody from an artificial immune system

interesting observation to emerge from this work is the fact that selection pressure acting only on the phenotype is capable of acting on the genotype. This suggests that the model might be extended to more complex fitness functions and applications in which it is straightforward to calculate some measure of phenotypic fitness of a single antibody but more difficult to quantify the fitness of the entire immune system as a whole. So far however, the model appears only to have been used in studies of natural immunological processes, and not extended to other domains.

2.2.2.2 Emergent Fitness Sharing

The library based model just discussed uses one bit string to represent an entire immune system. An alternative approach is taken by [Smith et al., 1993] who propose a population based algorithm in which a genetic algorithm is used to produce and maintain multiple subpopulations of antibodies within the complete *population*. Therefore, an immune system in this case is represented by the entire population manipulated by the genetic algorithm, rather than a single individual in the population as in Hightower's work. An important difference between Smith's model and the library-based model that arises from this is that in Smith's model the bit string represents both the genes that code for an antibody *and* the phenotypic expression of the antibody itself, and thus is a further simplification of the natural immune system.

The algorithm proposed in [Smith et al., 1993] to evolve and maintain a diverse population of antibody niches is known as *emergent fitness sharing*.

1. Choose an antigen at random.
2. Choose a sample of size σ of the antibody population, at random and without replacement.
3. Each antibody in the sample is matched against the chosen antigen, using a match-function M to compute its match-score.
4. The antibody in the sample with the highest match score has its match score added to its fitness. The fitness of all other antibodies remains unchanged.
5. Repeat from step (1) for typically three times the number of antigens.

Interactions between strings are defined by a matching function which rewards more specific matches over less specific ones in an effort to capture the immune systems ability to distinguish non-self from self — this is accomplished by ensuring that the recognition is specific.

Using this model, they show that their binary immune system is capable of detecting common patterns in a noisy environment, and that using the GA with the emergent fitness sharing function, it is possible to maintain diversity within the antibody population. Thus their immune system captures one of the essential characteristics of the natural immune system that it is capable of recognising an enormous number of foreign pathogens using limited genetic resources. They also show that like the natural immune system, their model can perform feature detection. This is made evident by performing experiments in which the GA evolves an antibody population of identical antibodies which can match multiple antigens by detecting common schema. This kind of common feature detection is very useful to the natural immune system, for example it learns to recognise certain bacteria by identifying a common polysaccaride contained in the cell walls of many different types of bacteria. Finally, their experiments show that within the antibody population, multiple peaks corresponding to recognition of different antigens can be maintained, and that the size of those peaks is proportional to the bias within the antigen population. This is somewhat reminiscent of clonal selection within the immune system, in which the lymphocytes that best recognise an antigen are proliferated and hence increase in number.

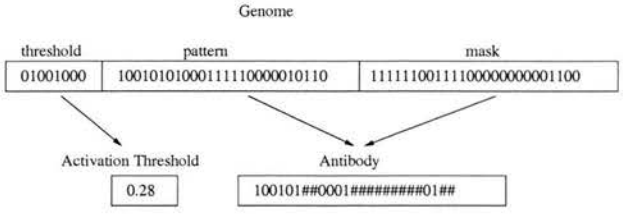

Figure 2.7: Mapping from B-Cell Genome to activation threshold and antibody (taken directly from [Potter and De Jong, 1998]

2.2.2.3 A Co-evolutionary Immune System

Another system comprising of an evolutionary algorithm working in conjunction with a binary representation of an immune system is described by Potter *et al* in [Potter and De Jong, 1998] and [Potter and De Jong, 2000]. In this model, an antibody representation is chosen which attempts to capture more closely a feature of the natural immune system in that it contains some antibodies which only recognise very specific antigen, and others that are more wide-ranging in their matching ability. The representation enables a spectrum of antibodies to be modelled, ranging in specificity from those that only bind to single antigen to those that match whole families of antigen sharing common characteristics.

The representation is shown in figure 2.7. As in Hightower's work, there is a mapping from genotype to phenotype to specify an antibody. The first 8 bits of the genome map to a real-valued activation threshold. The remaining part consists of a pattern and a map from which an antibody is generated. A mask bit of 1 generates a schema value equal to the corresponding bit in the pattern, a mask bit of zero produces a 'don't care' symbol in the antibody which matches anything. The mapping is many-to-one, i.e. many genotypes may result in the same phenotype. This is intended to represent another feature observed in real immune systems that many dissimilar chains of amino acids may fold into the same basic three-dimensional shape, and hence recognise similar antigen.

Rather than use the emergent fitness sharing algorithm just described in order to resolve the problem of preserving diversity within an antibody population, they propose the use of a co-evolutionary genetic algorithm in which individuals from multiple

non-interbreeding subpopulations collaborate to solve the target problem. The fitness of a B-Cell (and hence its antibody) is calculated by adding it to a 'serum' consisting of the best B-Cells from each of the other populations. The serum is then presented with a set of both self and non-self 'molecules' — the fitness of the serum is defined as the number of non-self or foreign molecules recognised by all the antibodies in the serum minus the number of self molecules recognised. Thus each B-Cell gains a reward that summarises how well it collaborates with other B-Cells to cover the collection of foreign molecules.

This system was applied to a concept learning problem, that of discriminating between the concepts 'Republican' and 'Democrat' by examining the voting records of members of the U.S House of Representatives. The performance of their immune system was compared to that of a symbolic inductive learning system AQ15, and the results showed that the immune system not only capable of learning the concepts, i.e. its predictive accuracy was equal to that of AQ15, but that the description of the concepts it produced was significantly more concise than that of AQ15.

The co-evolutionary architecture of this model in theory gives the potential for distributing the co-evolving populations to different nodes or machines, and hence confers robustness on the system, as there is no longer a single point of failure, and to this extent it is perhaps more faithful to the natural immune system model than the models of Hightower and Forrest described in the previous two sections. The model also contains other of the key characteristics of the natural immune system given in chapter 1, section 1.2, namely that the method of representing B-Cells from which antibodies can be derived allows feature extraction to be performed, and the utilisation of a threshold mechanism for detecting matching. Furthermore, the co-evolutionary architecture allows diversity to be maintained across the system. However, as the authors themselves point out in [Potter and De Jong, 1998], the model is of course an extremely loose model of an actual invertebrate immune system.

2.2.3 A Summary of EA Based Models

This section has presented several different models of immune system that incorporate an EA of one kind or another as a mechanism for evolving antibody sets with the

desired properties. Each of the models described operates in a binary antigen universe, and all the models exhibit a common subset of the features of the natural immune system, namely they perform recognition via probabilistic detection of pathogens, are capable of maintaining diversity, are able to learn the structure of the antigenic universe to which they are exposed, and to some extent are able to perform feature extraction. All the models draw inspiration from at least some features observed in the real system, for example the matching functions employed by Potter, Hightower, Forrest and Smith are all based on actual immunological observations.

None of the EA-based models explicitly makes use of the concept of memory detectors, although in the work reviewed none of the system had been applied to problems in which the environment is dynamic, hence the need to use memory detectors is perhaps unnecessary. On the contrary, in all the systems just described detectors are evolved to meet a specific goal, and once attained, the evolution is stopped. Thus, it could be argued that the memory detectors are merely the set of detectors or libraries that result from the evolution process. However, if detectors were required to be generated continuously as in ARTIS, an evolutionary approach could run into problems, due to the time-scales required to perform the evolution. There are two other key features of the natural system not exhibited by any of the EA models — self-regulation and co-stimulation. By definition, an EA must have a fitness function controlling evolution, and hence this can be considered analogous to having a central control function. Co-evolution, or the presence of a 2nd signal confirming the nature of the detection, is not incorporated into any of these models. Nevertheless, the EA seems to provide a sensible starting point for an artificial immune system, certainly in a binary universe, as it does provide a feasible method of searching the detector-space for suitable detectors, rather than randomly generating them as in ARTIS.

2.3 Network Models for Machine Learning

A number of implementations of artificial immune systems rely on the immune network metaphor. As previously mentioned, the network model of the immune system is disputed by some theoretical immunologists, never the less, significant progress has

been made in applying the idea to problems in machine learning. Two influential network models can be identified in the literature today — these are described in some detail, however first some of the background work that led to these models is briefly reviewed.

One of the earliest applications of the network idea to a machine learning problem was given by Cooke and Hunt, [Cooke and Hunt, 1995], who developed an AIS to classify sequences of DNA as promoter-containing or promoter-negative. This work attempted to closely adhere to the biological model — thus, for example it modelled B-Cells containing gene libraries and messenger RNA from which antibodies could be produced via a transcription mechanism, and it utilised matching rules weighted in favour of contiguous matching regions. B-Cells were stimulated according to the algorithm given by Farmer in equation 2.1, and clones of B-Cells produced via somatic hypermutation. New clones were then integrated into the network. Whilst the work yielded some promising results, it was unable to perform as well as a previously published neural network approach to classifying the data. The model was improved in [Hunt and Cooke, 1996] in an attempt to build an immune system capable of case-based reasoning. The idea was that each B-Cell in the network would represent a case, and similar cases would be linked together via the network which was self-organising in nature. The system contained both specific and generalised cases, attempting to mimic the way that the natural system can generalise over infections. This model still exhibited some major limitations as far as application to real-world complex data-sets. In particular, many problems were associated with building the immune network — if the network was randomly initialised, it took a long time to build useful patterns within the network, and there was an extremely high overhead associated with insertion and deletion of nodes into and from the network, especially as the size of the network grew. Furthermore, attempting to mimic the method by which matching occurs in the real immune system proved too simplistic, and only applicable to binary data strings. Further work described in [Hunt et al., 1999] resulted in a new system named *Jisys* which addressed these problems and was used to detect patterns in a database containing information relating to mortgage fraud.

Building on the foundations laid by Hunt *at. al*, a sequence of improvements pre-

sented in [Timmis et al., 2000] has led to the emergence of a system originally named *RLAIS*, Resource Limited Artificial Immune System, and now renamed *AINE*, described in [Timmis and Neal, 2001]. This represents one of the most sophisticated and successful network models in the current literature. Timmis claims:

> this system is a major step forward in making artificial immune systems a viable contender for effective unsupervised machine learning and allows for not just a one shot learning mechanism but a continual learning model to be developed

AINE introduces the concept of the *Artificial Recognition Ball*, or *ARB*. A network consists of a number of linked *ARBs*, with links representing similarity between them. Similarity is calculated on the basis of the Euclidean distance either between two ARB cells or between a cell and an antigen. The network initially consists of a cross-section of the data to be learnt, with the remainder of the training data comprising the antigen set. The system contains a fixed number of B-Cells — the *ARBs* compete for the ability to represent these B-Cells, according to their current stimulation level. Stimulation of an ARB is determined by three factors; the primary stimulation of the ARB by antigen (i.e. the data), *ps*; the affinity of an ARB for its neighbours in the network, *nn*; and finally by how much is is suppressed by its neighbours, *ns*. The calculation of the exact stimulation level *sl* is given in equation 2.3, where a is the number of antigens an ARB has been exposed to, pd_x is the distance between the ARB and the xth antigen in the normalized data-space, and dis_x is the distance of the xth neighbour from the ARB.

$$sl = ps + nn - ns = \sum_{x=0}^{a}(1 - pd_x) + \sum_{x=0}^{n}(1 - dis_x) - \sum_{x=0}^{n}(dis_x) \tag{2.3}$$

B-Cells are allocated to ARBs, depending on their stimulation level, regardless of how many B-Cells are actually available. Then, the weakest B-Cells are systematically removed until the number of B-Cells allocated is exactly equal to the maximum available. This introduces competition between ARBs and provides a mechanism for achieving population control. Remaining ARBs are cloned and mutated according to their stimulation level, and the clones are integrated into the network if their affinity to other *ARBs* in the network is below some fixed threshold. This gives rise to a meta-dynamical system which eventually stabilises into a network that represents the

patterns within the data. The network is visualised in order to observe clusters. The algorithm was applied to the classic Fisher Iris dataset, and resulted in the three known clusters clearly appearing within the network within twelve iterations of the algorithm. Although the network undergoes perturbations, the clusters are still visible after three hundred iterations. The system requires tuning of three parameters: the threshold governing insertion of cells into the network, the number of resources allowed, and mutation rate which controls diversity. More details concerning setting and effects of these parameters are given in [Timmis, 2000a, Knight and Timmis, 2001].

Timmis claims that the mechanisms used by the algorithm were inspired by phenomena observed in the natural immune system. Thus, he suggests that it is reasonable to assume that the natural system must contain a finite number of B-Cells and cannot undergo exponential growth in the the number of B-Cells, and therefore it is reasonable to limit the resources within the artificial model. It is also suggested that the behaviour of *AINE* simulates the *metadynamics* of the immune network discussed in section 2.1.1 — *AINE* maintains a core network describing the training data, no matter how many times the training data is presented, although perturbations occur from iteration to iteration. Finally, the concept of ARBs is consistent with a view expounded by Perelson that the biological system consists of a finite number of antibodies which are representative of an infinite number of antigens based on a notion of *shape space*. This is the idea that each antibody can recognise all antigens that occur in a volume V_e surrounding the antibody, and that if an infinite number of antigens can be placed in each volume V_e then a finite number of antibodies can recognise all antigen.

A fundamental point that must be addressed in relation to this model (and other related network models) is to consider how far its performance may be limited by its use of Euclidean distance between points as a measure of their similarity. For example, consider the Fisher Iris Data used by Timmis to test the *AINE* network. Closer examination of this data reveals that in the usual set of 100 measurements used in training, the 8th item belongs to one class, whilst the 91st item belongs to another. However, when the Euclidean distance between all pairs of points is examined, the 8th item is closest to the 91st item, despite the items belonging to different classes. Therefore, a clustering algorithm based on Euclidean distance between points without any kind of

supervision *could not* correctly classify these items. In order to separate these items, some warping of the dimensions must be undertaken, which cannot be performed without supervision. No information could be found in the literature as to the percentage of items in the Fisher set correctly classified by the *AINE* algorithm, therefore despite the appearance of three distinct clusters within the data in the diagrams given for example in [Timmis and Neal, 2001], it would be interesting to compare the number of items correctly classified to that produced by other more established methods.

The second influential network model is a system named *aiNet*, due to De Castro and Von Zuben, and described in [De Castro and Von Zuben, 2000b, De Castro and Von Zuben, 2001]. The system was designed with the goals of data clustering and of filtering redundant data. In this model, the immune network is considered as an edge-weighted graph, not necessarily fully connected, composed of a set of nodes (cells) and node pairs (edges) which are assigned a weight or connection strength. The network is evolutionary in the sense that evolution strategies are used to control the network dynamics and plasticity, and also connectionist once a matrix of connection strengths is defined to measure the affinities between the network cells. As in the *AINE* model described above, network cells compete for antigenic recognition, and those successful undergo cell proliferation, whilst antibody-antibody recognition results in network suppression. Similarity in this system is also calculated on the basis of Euclidean distance between cells. The exact algorithm is given in figure 2.8 — steps $(1(a)i\text{-}1(a)vii)$ simulate the clonal selection and affinity maturation processes occurring in the natural immune system, steps $1(a)(viii)\text{-}1(a)(x)$ and steps $1(b)\text{-}1(c)$ simulate the metadynamics of the immune network. Note that steps $1(a)(ix)$ and $1(a)(x)$ introduce both clonal suppression and network suppression elements to the algorithm. The model contains rather a large number of parameters, and also has a high computational cost per iteration (of the order $O(p^3)$). Furthermore, it is difficult to determine sensible stopping criteria.

The network outputs consist of a matrix of memory cell coordinates and a matrix of inter-cell affinities. The network structure is analysed by calculating the minimum spanning tree of the network — this allows the clusters in the data to be identified and also a means of determining which network cell belongs to which cluster. Results are

reported on the application of *aiNet* to two problems — a simple data-set consisting of 5 linearly separable clusters, and the well known two-donut problem. For the first problem, clusters are correctly identified, and the algorithm produces a 66% compression rate. For the donut problem, again correct classification is achieved alongside a reduction in the dataset size of 92%.

In summary, *aiNet* and *AINE* both draw heavily on the immune network metaphor in order to produce systems which are well adapted to clustering real-valued data, and incorporate many of the characteristics of the immune system outlined in section 1.2 of chapter 1. Their drawbacks lie mainly in the high overhead of maintaining the dynamical network by inserting and removing nodes, which may prove a significant deterrent to applying the network to rapidly changing datasets. This is obviously an intrinsic disadvantage of any network based model. Another factor to note is that secondary visualisation processes have to be applied to interpret the networks produced and it is not straightforward to determine the class of a new item of data. Finally, as already noted above, the use of Euclidean distance as a similarity measure may limit the ultimate effectiveness of network based models. It seems possible that artificial data-sets could be constructed to serve as a counter-example in which data could *not* be clustered by either the *AINE* or *aiNet* algorithms.

2.4 Supervised Learning Using An Artificial Immune Model

The network models just presented that perform data-clustering are *unsupervised* learning systems — this thesis is also concerned with unsupervised data-clustering. However, it is worth examining the features present in a supervised learning system proposed by [Carter, 2000]. This supervised learning algorithm based on an immune system analogy performs pattern recognition and classification. The system is known as *Immunos* – 81, and uses abstractions of T-Cells, B-Cells, antibodies and their interactions. Artificial T-Cells control the creation of B-Cell populations or clones, which compete for recognition of unknown data-items. The clone with the highest affinity for the data is then said to recognise and thus classify the unknown data.

1. At each iteration step, do:
   (a) For each antigen i, do:
      i. Determine its affinity to all the network cells according to a distance metric in shape-space, d_{ij}
      ii. Select the n (or $n\%$ of the) highest affinity network cells;
      iii. Reproduce (clone) these n selected cells. The number of progeny of each cell, N_c, being proportional to their affinity: the higher the affinity, the larger the clone size;
      iv. Increase the affinity of these N_c cells to antigen i, by reducing the distance between them (corresponding to greedy search)
      v. Calculate the affinity of these improved cells with antigen i
      vi. Re-select $\tau\%$ of the most improved (highest affinity) cells and put them into a partial matrix M_p of memory cells
      vii. Eliminate those cells whose affinity is inferior to threshold d (affinity threshold), yielding a reduction in the size of the M_p matrix
      viii. Calculate the network cell-cell (Ab-Ab) affinity, s_{ij}
      ix. Eliminate those cells whose affinity s_{ij} is inferior to threshold s, leading to another possible reduction in M_p (clonal suppression);
      x. Concatenate the original network cell matrix with the partial matrix of memory cells ($C \leftarrow [C;Mp]$)
   (b) Determine the whole network inter-cell affinities and eliminate those cells whose affinity with each other is inferior to threshold s (network suppression)
   (c) Replace $r\%$ of the worst individuals by novel randomly generated ones
2. Test some defined stopping criterion and stop if necessary

Figure 2.8: The aiNet algorithm (taken from [De Castro and Von Zuben, 2000b])

Immunos – 81 embodies the biological concepts of T-Cells (in contrast to most AIS models reviewed in this chapter), B-Cells, learning, and recognition. T-Cells are used to control the production of B-Cells within the system, and learn the primary structure of antigens, that is in this case the type and sequence of variables in the antigen data. B-Cells in the system perform 'instance recognition', i.e. they recognise specific features of each individual antigen. The system does not explicitly represent a concept of self, neither does it simulate idiotypic interactions between B-Cells. Furthermore, it does not contain any pre-formed T or B-Cells, they are created as the system runs.

The system was tested on two standard machine-learning data-sets, consisting of eight nominal and six continuous variables. The first data set known as the Cleveland data-set consisted of 303 records from patients with suspected coronary artery disease. This was used as a training set prior to presenting the system with 200 unknown patient cases. Correct classification of the unknown data was produced in 80.3% of cases, and when cross-validation runs on the original Cleveland data set were performed, performance ranged from a high of 96% to a low of 63.2%. These figures compared very well with other machine learning techniques, the closest competitor being a *k*-nearest neighbour classifier.

2.5 Immune Algorithms for Scheduling

As stated in the introductory chapter, one of the themes of this thesis is to explore the use of an immune metaphor in the context of scheduling. An extensive search of the literature revealed some other applications of artificial systems to this broad domain. [Fukuda et al., 1999, Mori et al., 1997] describe a general framework for an autonomous distributed system to control semiconductor production. Their system consisted of multiple agents, corresponding to features of the immune system, to control production. Thus, the framework consisted of four types of agent, detector agents, mediator agents, inhibitor agents and restoration agents, which interacted with each other and with a production line, Detector agents,corresponding to B-Cells in the immune system, were used to detect specific malfunctions in the system. Mediator agents mimic the behaviour of lymphokines which are secreted by T-Cells in the immune sys-

tem to activate B-Cells. The function of these agents was to activate an inhibitor agent to inhibit firing. The inhibitor agents corresponded to B-Cells killing antigens, and the restoration agent (corresponding to helper T-Cells in the biological system) served a purpose similar to the detector agents. The framework has not been tested in practical applications but the authors claim that the framework could enable decision making in real-time and tolerance to a changing environment. A similar agent-based immune system was described by [Russ et al., 1999] for performing task allocation in computer systems, in order to make a system capable of adapting to a changing environment.

Of more interest to the type of scheduling that it is proposed to tackle in this thesis is work by [Mori et al., 1998] and [Costa et al., 2002]. Mori *et al* describe an AIS which is used as an optimisation algorithm to solve a multi-objective scheduling problem. The problem they discuss is as follows: orders can be split into several jobs of suitable batch sizes. Each job is then processed on a sequence of machines. Thus, two sub-problems exist: the first is a job-splitting problem in which each order consisting of splittable jobs must be split into optimal batch sizes, and the second is then a standard job-shop scheduling problem which determines the sequence that each job is processed on each machine.

Mori *et al* propose an immune algorithm that mimics the idea of an immune network as proposed by [Jerne, 1973], in combination with somatic recombination and mutation in order to maintain diversity in the antibody population. Their system produces schedules for single problems in which a set of objective functions is optimized, and therefore does not consider building in robustness and flexibility into the schedule to cope with unpredictable events, as discussed in section 1.3.1 of Chapter 1. However, the authors propose that their system could be adapted in future to cope with dynamical environments in which orders of jobs are changed or the objective functions vary dynamically.

The immune network consists of two types of antibody — one which encodes batch sizes (as integers), and a second which encodes the job priority, and is a permutation of integers. An antigen represents the conditions of the problem, such as the order quantity and objective functions. Affinity between two antibodies is defined by measuring the *informative entropy* between the antibodies — this quantity is a measure of the

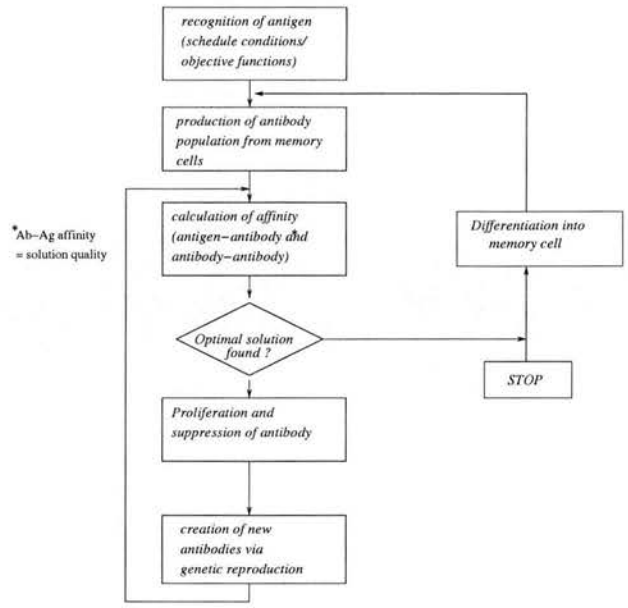

Figure 2.9: Immune Algorithm for Production Scheduling

diversity between the antibodies, measured by comparing allele values at each locus. Affinity between an antibody and antigen is simply the value of the objective function given by solving the problem defined by the antigen using the batch-sizes and sequence defined by the antibody. Antibodies proliferate and are suppressed depending on their concentrations and affinities for other antibodies in the network and for the antigen. Those antibodies with high affinity for antigen (and therefore representing low cost solutions) proliferate — however, if the concentration of an antibody becomes excessive, its proliferation is suppressed, in order to enable exploratory search as well as focusing on local minima. If an antibody is suppressed until it is eliminated, new replacement antibodies are generated by genetic reproduction operators such as crossover and mutation. When a good solution has been found, it is stored in the memory cells so that it can be immediately retrieved if the same antigen is presented in the future. The operation of the system is shown in figure 2.9.

Compared to a straightforward 'generate and test' method, the immune algorithm provided better results with respect to waiting time, which is claimed to be due to the immune algorithm's ability to search the global space of solutions. Essentially, the

immune system is being utilised as an optimisation algorithm in these cases. Although the algorithm itself is faithful to the network model of immunology, the manner in which it is applied to the problem area does not take full advantage of the potential of the analogy for tackling practical features of scheduling problems.

[Costa et al., 2002] introduce an immune algorithm for make-span minimisation on parallel processors. This algorithm is also an optimisation algorithm, in that it finds an optimal solution to a single, specific scheduling problem, and does not consider the question of producing solutions which are robust to changes in the environment. However, the algorithm is described for completeness. The immune system is based on the *CLONALG* algorithm due to De Castro, [De Castro and Von Zuben, 2000a]. A simplified version of this algorithm is shown in figure 2.10. Antibodies consisting of strings of integers represent feasible solutions to the scheduling problem, and the affinity of each antibody is given by $\frac{LB}{1+M(k)-LB}$ where LB is a derived lower bound for the problem, and $M(k)$ the make-span of the antibody under consideration. The process is run until no improvement in the best solution can be found. The algorithm was tested on 390 instances of generated problems in which each instance had i processors on which to schedule j jobs, with the processing time of each job obeying a uniform distribution in the range $[1,k]$. They compared their results to tabu search, simulated annealing, local search, and a number of heuristics. Their results were encouraging, showing that the immune based algorithm was effective in dealing with instances characterised by jobs with long processing times on a small number of machines, especially when compared to single-solution strategies. It is claimed that these superior results are due to the ability of the immune system approach to maintain diversity within a population of candidate solutions.

2.6 Artificial Immune Systems for Dynamic Problems

One of the key characteristics of the immune system is its ability to function in a dynamic environment. In this section, applications of artificial immune systems which directly exploit this analogy are considered. [Gaspar and Collard, 1999, Gaspar and Collard, 2000] have applied an immune system analogy to producing a

1. Create a population of k antibodies representing feasible solutions to the problem
2. For each generation do:
   - For each antibody do:
     - decode the antibody
     - determine the antibody affinity
   - determine the number of clones of each antibody
   - determine the number of mutations of each antibody
   - do cloning and mutation
   - For each clone do:
     - decode the clone
     - determine the clone affinity
     - if Affinity(clone) > Affinity(antibody), replace antibody with clone
3. whilst stopping criteria not met

Figure 2.10: The CLONALG algorithm [De Castro and Von Zuben, 2000a]

system to perform time-dependent optimisation. They consider a canonical benchmark problem, in which the fitness of a solution is defined by its similarity to an arbitrary bit-string which represents the current optimum and changes regularly. The problem can be parameterised with respect to the continuousness of transitions, the transition period, and transition range, and hence makes a suitable benchmark. Gaspar *et al* propose an immune based algorithm capitalising on the biological immune systems adaptive nature. An antigen represents the current optimum, which must be matched by B-Cells. Their system, named *Sais*, features both a primary (reactive) response and a secondary response, mimicking memory. The system starts with a random population of B-Cells, each able to detect a given antigen. At each generation, three operators are applied to the population; evaluation, clonal selection and recruitment. The *evaluation* phase results in an *exogenic* activation for each B-Cell, based on the Hamming Distance between the B-Cell and the current optimum, and an *endogenic* activation based on the number of different types of B-Cells in the population and the current density of the B-Cell itself in the current population. This tends to force convergence of the system towards a set of equally represented categories. The *Clonal Selection* phase produces intermediary populations of both exo-activated and and endo-activated B-Cells, by applying a selection operator according to their activation level, and in the case of exo-activated cells, applying somatic hypermutation. Cells from both intermediary populations are then combined into a new population in the *recruitment* phase, again using a selection process. Results show that *Sais* is capable of handling the dynamic optimisation problem described. The authors attribute its success to the fact that their system is *reactive* in the sense that it can discover new optima, but also that it can preserve diversity (unlike, for example, straightforward genetic algorithms) so that it can remain reactive over time. The model is loosely based on the idiotypic networks presented in the previous sections, however is clearly an over-simplified model of the immune system itself. Nevertheless, it captures some of the essential mechanisms by which the immune system operates in a dynamic, complex environment, and exhibits analogies of the primary and secondary responses due its maintenance of multiple categories of B-Cells within the population. This work also reinforces the notion that an artificial immune system may be a convenient metaphor to work with in non-stationary

environments.

The preceding sections have presented an overview of the main techniques used to implement artificial immune systems. Furthermore, a detailed summary of the literature relating to the topics of interest in this thesis has been given. The field is expanding rapidly however, and there are many more examples of AIS in the literature than could be described in this thesis. The interested reader is referred to a detailed bibliography containing 293 references produced by [Dasgupta et al., 2002] — this bibliography contains a wide variety of applications, implemented using variations on one or more of the techniques described in this chapter.

2.7 Sparse Distributed Memories and their Relationship to Immunological Memory

Smith *et. al* have shown that immunological memory is a member of a class of sparse and distributed associative memories. Another type of memory typical of this class is Kanerva's Sparse Distributed Memory, or SDM [Kanerva, 1988]. The work presented in chapters 5 and 6 of this thesis draws heavily on the analogy between the SDM and the immune system, and hence a brief description of the SDM is now presented to outline the underlying concepts so that the correspondence between the two types of memory can be made clear. A more detailed discussion of the properties of the SDM is provided later in this thesis in chapters 4 and 6.

2.7.1 Kanerva's model

The SDM is a form of memory which can be written to by providing an address and data, and then read from by providing an address and getting an output. The SDM is specifically designed to function with enormous address spaces, in which it would be impossible to physically instantiate all of the possible address locations. For example, SDM can cope with addresses of 1000 bits, and therefore 2^{1000} potential address-data locations. An SDM instantiates a small and random subset of these locations, which are referred to as hard locations, and are said to sparsely cover the input space

[Kanerva, 1988].

Each hard location has an associated set of *counters*, one for each bit. Whenever an address is presented to memory, the Hamming Distance between the address and each of the hard locations is calculated. All hard locations that are within some threshold distance R, referred to as the *recognition radius*, of the address become active — this subset is called the *access circle* of the address. The method in which read and write operations are performed is shown in figures 2.11 and 2.12.

In order to *write* an item of input data to the memory, each bit of input data is stored at every location in the access circle: if the ith input bit is 1, then the ith counter at each hard location in the access circle is incremented by 1, if the ith input bit is 0, then the ith counter at each hard location in the access circle is decremented by 1.

To *read* an item from the memory, the sum of the ith counter value of each of the hard locations in the access circle is calculated, for each bit: if the sum of the ith counters is positive, the ith output bit is 1, if the sum of the ith counters is negative, the ith output bit is 0.

The SDM has several appealing properties. The data is distributed independently to many hard locations, thus it is robust to the loss of individual hard locations and exhibits a graceful degradation of response. Furthermore, the mechanism by which data is retrieved is imprecise, therefore data can be retrieved even if a read address is corrupted and hence slightly different from a prior write address, due to the associative nature of the recall. The associative behaviour results from the overlapping of access circles — if the access circle of a read address overlaps that of the write address, then all locations within the overlap are activated, and thus give associative recall.

2.7.2 Correspondence between SDM and Immunological Memory

Smith *et al* discuss the correspondence between SDM and immunological memory (IM) in great detail in [Smith et al., 1999]. This section reiterates their argument to illustrate the close relationship between the two systems. Table 2.1, reproduced from [Smith et al., 1999] summaries the correspondence between the two memories.

Both IM and the SDM perform recognition by means of detectors; in the case of SDM, the recognition is of addresses, via hard locations. In IM, the recognition

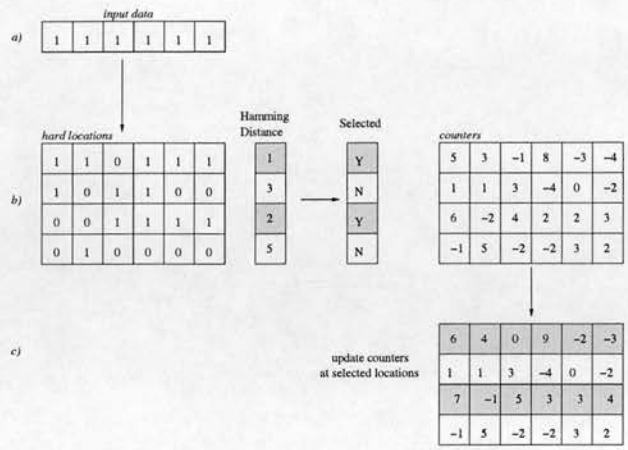

Figure 2.11: *Writing* a piece of data to the SDM with a recognition radius $R = 2$: a) The input data is presented to the memory. b) The Hamming Distance D between the data and each hard location is calculated, and those locations in which $D \leq R$ are selected. c) The counters at each bit of the selected locations are updated according to the input data

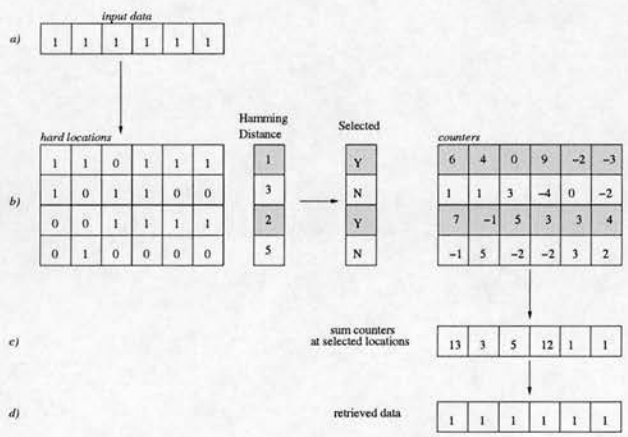

Figure 2.12: *Reading* a piece of data from the SDM with a recognition radius $R = 2$: a) The input address is presented to the memory. b) The Hamming Distance D between the data and each hard location is calculated, and those locations in which $D \leq R$ are selected. c) The counters at each bit of the selected locations are summed d) Bits in which the sum ≥ 0 output 1, bits where the sum is < 0 output 0

Immunological Memory	Sparse Distributed Memory
Antigen	Address
B/T Cell (Antibody)	Hard Location
Ball of Stimulation	Access Circle
Affinity	Hamming Distance
Response/Tolerance	Data
Primary Response	Write and Read
Secondary Response	Read
Cross-Reactive Response	Associative Recall

Table 2.1: Structural and functional correspondence between immunological memory and SDM. The table is taken directly from [Smith et al., 1999]

is of antigens, by B/T cells which produce antibodies. In both cases the potential recognition space is huge, and therefore both systems can only sparsely cover their respective input spaces with detectors. Thus, in both systems, recognition can only be imprecise, and thus detectors become activated if they are within some threshold distance of the input. In IM, the threshold is determined by the binding affinity between antigen and antibody, in SDM the threshold is determined with respect to Hamming Distance. In either case, a subset of detectors becomes active on presentation of an input, in the case of IM, this is called the ball of stimulation of the antigen, in SDM it is referred to as the access circle.

Both systems store information associated with each input. In the case of SDM, the information is simply contained in the bit-strings comprising the input data. In IM, the equivalent information stored is determined by the immune response itself, i.e. the mechanism by which the immune system responds to an antigen and the types of cells invoked to perform that mechanism.

Both systems perform associative recall. In an SDM, if a noisy or corrupt input datum activates a set of detectors that overlap with those activated by a prior input, then detectors from the prior input will contribute to the output. In IM, a mutant strain of an antigen plays the same role as the noisy input in SDM, and a previously activated

antibody will contribute to the response.

Thus, it should be clear from the preceding discussion that the SDM and immune system analogies are interchangeable, and that elements from both models could theoretically be incorporated into any new AIS model.

2.8 Conclusion

This chapter has presented a broad overview of some basic immunology and of the diverse range of artificial models of the immune system currently described in the literature. The models range in nature from those that attempt to closely model biological phenomena, to those that simply use the metaphor as loose inspiration for a computational system. All of the systems described exhibit a subset of the features of the real immune system described in chapter 1. The remainder of this thesis introduces four new models based on the immune analogy, which also attempt to capture the salient characteristics of the immune system. The problems domains to which they are applied have been chosen to capitalise on these characteristics as much as possible.

Chapter 3

Immune Systems for Scheduling

3.1 Introduction

The previous chapters have outlined the properties of the biological immune system that would seem important in an information processing context in some detail, and provided an overview of a number of computational implementations of systems which incorporate various subsets of these properties. The discussion of these properties (in both biological and artificial terms) implies that the immune system metaphor would lend itself most readily to those type of real-world applications which operate in dynamic and unpredictable environments, in which it is necessary to react to changes in a timely manner and in which memory of past experiences provides useful pointers as to the correct course of action. The biological immune system is not perfect; not every potential pathogen can be recognised by every individual, and those that are recognised often recognised imprecisely. Despite this, the system as a whole functions sufficiently well for the human race to have survived for many thousands of years. This suggests that potential application areas for the application of the metaphor are those in which we are not seeking to find *optimal* solutions or answers to some problem, but more simply robust and 'good enough' solutions that allow the system to continue operating.

Some example application areas fitting these aspirations have already been described in chapter 2. In the introductory chapter, it was suggested that another ob-

vious application area in which many of the characteristics describing the environment in which the IS operates are paralleled is that of *scheduling*. This analogy was made explicit in section 1.3.1 of chapter 1. Here, it was noted that an 'ideal' schedule is not necessarily one which optimises some measurable criterion such as make-span or maximum tardiness, but one which has some built-in flexibility that can absorb some unpredictable event without disrupting the planned schedule. Also, it was made clear that the schedule must still deliver some acceptable level of quality when measured against some pre-determined criteria. This chapter first provides an overview of current approaches to the re-scheduling problem identified in the (non-immune system) literature. The job-shop scheduling problem is then properly defined so that the analogy between a scheduling system and the immune system can be made explicit (section 3.4). Two approaches to immunology based scheduling are then described.

3.2 Other Approaches to Robust Scheduling

The type of real problem described in the introduction has received much attention over the years from many areas of the academic community, initially from the the Operations Research community and later from Artificial Intelligence, using a wide and varied list of techniques. The real-world scenario described is often typified by the *job-shop* scheduling problem, of which there exist many benchmark examples on which algorithms can be compared, (for example [Beasley, 1990]). These problems are generally NP-hard and cannot be solved to optimality — however, a great deal of attention has been paid to attempting to produce schedules that are as close to optimal schedules as possible, in the sense that some objective is minimised. In the majority of cases however, such 'optimal' schedules are often extremely fragile — a minor change in conditions can render the schedule useless. This has no practical value in a real world situation, in which if any rescheduling is required, it is often desirable to produce a new schedule which resembles the old one. For example, a new schedule which differs significantly from the original one may result in having to recall employees from holiday or changing set-ups on machines, both of which have economic implications.

A review of the literature indicates that comparatively little attention has been paid

to this problem of rescheduling. There appear to be two possible approaches:

1. Reschedule from scratch from the point at which a breakdown occurs, thus considering a completely new scheduling problem.

2. Produce a schedule in the first place that is capable of absorbing changes in the environment.

The approach outlined in (1) above has been adopted by [Bierwirth et al., 1995, Fang et al., 1993], using genetic algorithms. Approach (2) was taken by [Wu et al., 1999] though this work is in respect to making schedules robust to disturbances in operation processing times, and does not consider other possible events that may perturb the original schedule. This work employed a graph-theoretic approach combined with a branch-and-bound algorithm. [Herrmann, 1999] used a co-evolutionary algorithm to tackle a similar problem in which the algorithm converges to the worst case and therefore most robust scenario. [Jensen and Hansen, 1999] propose a new method of creating robust solutions to job-shop problems that produces solutions which are robust to breakdowns of machines.

In this chapter, a new method of producing robust schedules to job-shop scheduling problems is proposed which falls into category (2) above. Thus, it is proposed that an immune system analogy can be used to construct schedules in a manner which produces schedules that are resilient to changes in the environment. The use of the immune system analogy also results in a scheduling system that contains sufficient 'building blocks' to potentially construct schedules that cover a wide range of contingencies.

3.3 Definition of the Job-Shop Scheduling Problem - JSSP

In a typical job-shop scheduling problem (JSSP) j jobs are required to be scheduled on m machines. A single plan defines the $(j * m)$ operations that must occur. Each operation has a fixed processing time p_{jm}. Each job is expected to arrive at the factory

at time A_j, and must be completed by due-date D_j. A machine can only process one job at a time, and preemption of any operation on any machine is not allowed. In both models proposed in this chapter, we only consider contingencies involving unexpected arrival dates of jobs into the factory — deviations in arrival date can result in jobs having to be stored ready for processing for long periods of time if they arrive early, or cause delays in processing of other jobs if they arrive late. However, the methodology described is sufficiently generic that other contingencies such as deviations in due-dates or processing times could be incorporated straightforwardly.

There are many objectives by which the quality of a schedule can be measured, based on either completion time or due-date. Those based on completion time, for example the total production time (makespan), are rarely of commercial interest as the exact details of a problem are not known from the start and they do not have a clear end. In the case of job-shop scheduling, it is more sensible to consider objectives based on the date by which individual jobs are supposed to be completed. In this work we opt to use the measure of *maximum tardiness*, T_{max}, with the implication that a schedule's cost is directly related to the latest job that completes after its due-date. If each job j completes at time C_j, then the maximum tardiness of the schedule is defined as shown in equation 3.1, and means that the cost of a schedule is directly related to the amount of time that the latest job completes after its due-date.

$$T_{max} = max(0, C_j - D_j) \tag{3.1}$$

Other due-date based objectives, such as weighted tardiness or number of tardy jobs are simply variations on this theme.

3.4 Definition of an Immune-Based Scheduling System

A stylised model of the environment in which an immune-based scheduling system might operate is shown in figure 3.1. The relationship between the standard immunological terms and a scheduling environment is defined below:

Antigen — a set of conditions describing a possible scenario in the factory for which a schedule must be produced, i.e. each antigen defines one possible set of arrival

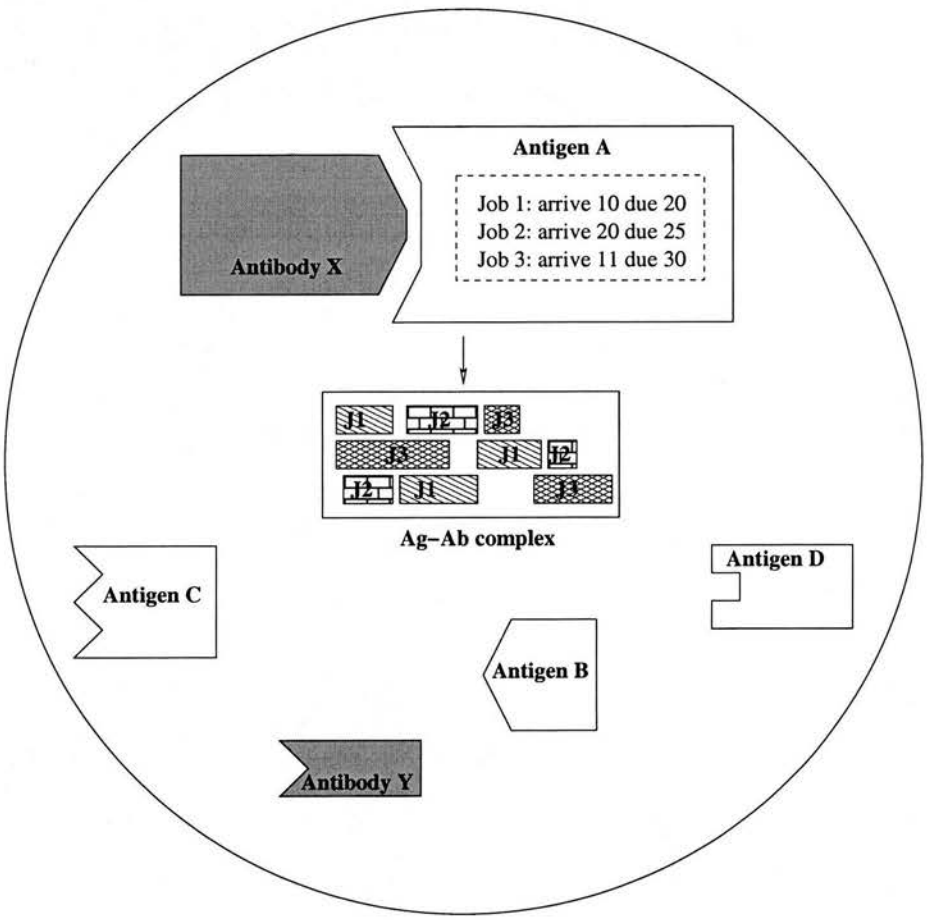

Figure 3.1: A stylised model of an Artificial Immune System for scheduling

dates and due-dates under which a schedule must be produced.

Antigen Universe — a set of antigens representing a sample of the possible scenarios and contingencies that may occur.

Antibody — a set of instructions for constructing a schedule.

Ag-Ab Complex — combination of an antibody and antigen produces an Ag-Ab complex which in this case represents a complete schedule, with start times and finish times for each operation. The schedule is produced using the instructions given by the antibody and the information supplied by the antigen.

Match-Score — the match-score represents the strength of the Ag-Ab complex, i.e. in a scheduling scenario the maximum number of time units a job is late in the schedule produced as a result of matching antibody and antigen. A perfect schedule has a match-score of 0 — the higher the match-score, the worse the schedule.

Thus, the scheduling immune system model contains a set of antibodies defining possible methods for creating schedules, and a set of antigens, representing a sample of the potential situations that may arise. The number of antibodies is small compared to the number of potential situations, therefore each antibody should match (to a greater or lesser extent) a *subset* of the antigens in the universe. Thus, there are three obvious and important questions to address when designing such an immune system:

1. How can one represent both antigens and antibodies ?
2. What are the building blocks from which the immune system assembles antibodies, and how are they generated ?
3. How are antibodies assembled from these building blocks ?

The next section proposes a model — $SCHED1 - IS$, based on immunological principles, that can accomplish these tasks, and discusses the reasoning behind the design choices that were made.

3.5 $SCHED1-IS$

3.5.1 Choice of IS Model

Having established precisely those features of the biological immune system which we wish to incorporate into an immune-based scheduling system, returning to the published literature showed that there were two obvious candidates of artificial IS models on which a system could be based. The first of these models, proposed in [Hightower et al., 1995], was described in detail in chapter 2, and a further model due to [Oprea and Forrest, 1998] is described below. Although both model binary universes in which bit-strings represent both antibodies (genotypically and phenotypically) and antigens, in principle, both could be extended to include more complex representations.

To recap, [Hightower et al., 1995] described a binary model of the immune system which was used to study the effects of evolution on the genetic encoding for antibody molecules, which showed that robust pattern recognisers can be learned with a surprisingly small amount of information. In this model, each individual in a population manipulated via a genetic algorithm represents the genetic specification for the antibody libraries of one immune system. Bit strings were used to represent both the genotype — libraries of gene segments — and the antibody molecules of the phenotype. Further work ([Perelson et al., 1996]) provided insight into how and why the natural IS evolved as it did.

Oprea in [Oprea and Forrest, 1998, Oprea and Forrest, 1999] describes a simplified version of the Hightower model which was used as part of a detailed study on the sources and evolutionary significance of diversity in the biological immune system. In this model, one individual's genome consists of a *single* library containing a set of *complete antibodies* — i.e. the single antibody library represented by an individual can be viewed as exactly the antibody repertoire. It is claimed in [Oprea and Forrest, 1999] that this does not affect the generality of the model and adding more libraries would not affect the results. The aim of the work was to investigate the type of antibody repertoire that might evolve in relation to a given pathogenic environment.

We opted to extend the more general approach taken by [Hightower et al., 1995],

using multiple libraries, as it appeared to fit the scheduling analogy and philosophy we were trying to create more closely; in a scheme containing multiple libraries, each library can be viewed as holding a piece of a schedule "jigsaw". Selecting different pieces from each library and combining them into a whole schedule allows for the creation of many different schedules. In contrast, if Oprea's approach were adopted, a library would contain *completed* schedules, therefore the whole library is simply offering a limited selection of alternative schedules, and thus is less flexible.

3.5.2 Representation of antibodies and gene-libraries in $SCHED1-IS$

Having elected to represent the scheduling system as a series of libraries of components which are randomly selected from, and recombined into a complete schedule, the immediate difficulty to be faced is exactly how to represent the schedule; the representation must be of a form that can be broken down into random fragments that can be recombined in a manner which *always* guarantees a feasible representation of a schedule.

As mentioned previously, the majority of AIS models operate in a binary universe. [Nakano, 1991] have described a binary representation to encode schedules for academic job-shop problems. However, a complex effort was required to design such an encoding, and its use in a GA context needed specialised repair operators to retain the ability to decode chromosomes as feasible schedules, therefore the representation is not suitable for use in the library-based model we propose.

However, an *indirect* representation of a schedule in which the representation encodes the *method* for constructing a schedule rather than the schedule itself offers obvious advantages, in that it can generally be manipulated by standard genetic operators without loss of meaning. A search of the relevant GA literature revealed one such representation which possessed the ideal properties. This representation was proposed in [Fang et al., 1993] and consists of a string of length $j \times m$ integers which is interpreted as follows:

> If the string is represented as "abcd..." then place the 1st untackled task of the ath uncompleted job into the schedule in the first place where it will

> fit, then place the first untackled task of the bth uncompleted job into the schedule etc.".

This representation was found to be very successful in tackling a wide range of scheduling problems at the time of publication, though since this work was completed it has been superseded by an alternative indirect representation proposed by the author of this dissertation in [Hart and Ross, 1998]. However, it remains suitable for the purposes of building and evaluating an immune system model. Clearly, a string representing a set of schedule building instructions in this manner can be broken down into sub-fragments which can be combined in any manner.

Thus, $SCHED1-IS$ is composed of a set of l libraries, each containing c components. Each component is a string of s genes, subject to the constraint that $s \times l = j \times m$, i.e. so the length of the expressed antibody must be equal to the number of operations. Each of the s genes has a value in the range $(0-(j-1))$. c can be varied independently. This is shown in figure 3.2.

3.5.3 Representation of an Antigen

An *antigen* describes a set of expected arrival dates and due-dates for each job in the shop and hence each antigen represents one of the contingencies we wish to deal with. Therefore, each antigen is simply an ordered list of dates represented by integers:

```
        Arrive   Due
Job1:   10       20
Job2:   5        11
Job3:   15       17
```

Note that information regarding the sequence in which each job visits each machine is separately maintained as this information is constant in the system. The antigens therefore represent only the variable information in the system. Of course, if the sequencing of jobs was also considered to be variable in the system, then the antigens could also represent this information.

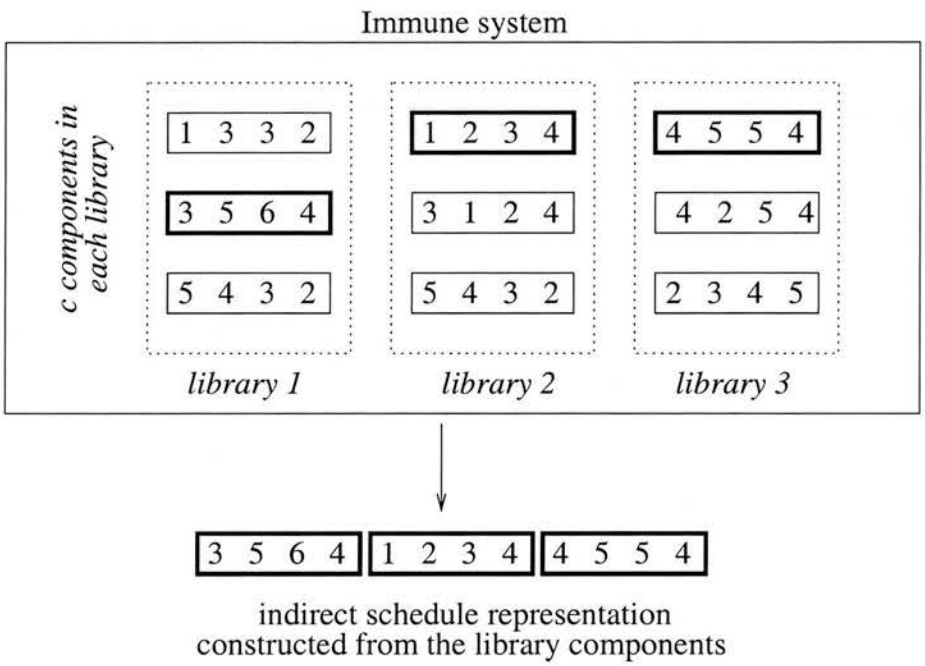

Figure 3.2: The figure shows an example of the immune system represented in $SCHED1-IS$. The immune system consists of l libraries, each containing c components. Randomly selecting one component from each library and concatenating them produces an indirect, feasible representation of a schedule

3.5.4 Evolution of the Gene Libraries

A genetic algorithm is used to evolve a set of immune libraries as shown in figure 3.2 in exactly the same manner as [Hightower et al., 1995]. Each individual in the population represents a complete set of libraries, i.e. an entire immune system. As in [Hightower et al., 1995], a haploid representation is used in which the total number of genes in each individual is $(l \times c \times s)$. Each *AIS* in the initial population is generated by assigning a random value to each gene. The fitness of an individual is determined by its overall ability to produce schedules which optimise T_{max} across all the potential scenarios described in the antigen universe, i.e. against all potential antigen encounters.

The procedure by which fitness is calculated is a modified version of that given in [Hightower et al., 1995], as described in chapter 2, section 2.2.2.1. A set of antibodies (schedules) are expressed from an individual by combining one component from each library (see section 3.5.2 of this chapter) and then exposed to the antigen universe. For each antibody-antigen encounter, a schedule is constructed, and the quality of the schedule in terms of T_{max} measured. Each antigen receives an *antigen-score* which is the minimum, i.e. the best, of all the values of T_{max} measured for that antigen. The overall fitness of an individual is computed by averaging all the antigen-scores; this assumes the survival probability of an individual depends on *all* the pathogenic challenges it encounters. An alternative approach would be to take the view that in fact the survival probability is dependent on the extent to which it is able to deal with the most difficult encounters, and therefore its fitness is characterized by the *worst* value of T_{max} found during an antigen encounter. The exact algorithm for computing fitness is given in figure 3.3.

After a match-score for an antibody has been calculated, the antibody is mutated at random in M positions, and the match-score recalculated. If the match-score improves, then the original antibody is assigned this new match-score. The mutations are not written back to the gene library.

1. Express N antibodies at random from each individual
2. Select K antigens at random, with replacement, from the antigen universe.
3. For each of the K antigens selected:
   - Using the arrival-dates defined by antigen K_i, produce N schedules, using the N expressed antibodies.
   - Calculate T_{max} for the each of the N schedules
   - Apply somatic mutation to each antibody and re-calculate the value T'_{max} for each schedule.
   - Assign antigen K_i an *antigen-score* equal to the best (i.e. lowest) value of (T_{max}, T'_{max}) found
4. Average the K antigen-scores to give an overall fitness for the individual

Figure 3.3: Algorithm for computing the fitness of each individual in $SCHED1-IS$

1. Express N antibodies at random from a set of evolved libraries
2. For a given antigen A, calculate the quality of the schedule generated (T_{max}) by applying the instructions encoded in each of the N antibodies
3. Select the antibody Ab^* with the best value of T_{max}
4. Produce C clones of Ab^*, by mutating each gene with probability p_m.
5. Calculate the new value of T_{max} for each clone, and return the best, C^*.

Figure 3.4: Algorithm for simulation of the immune-response. This algorithm produces a schedule in response to a change in scheduling conditions

3.5.5 Evaluating the IS Produced — Inducing an Immune Response

[Hightower et al., 1995] showed that the method just described allows a genetic algorithm to optimise complex genetic information, even though selection pressure is acting on the phenotype which expresses incomplete genotypic information. However, we are concerned with producing a *practical* scheduling system. This means that once a set of immune libraries has been evolved, we must be able to generate high quality schedules from those libraries in response to a new scheduling scenario quickly and efficiently. Thus, we must model and evaluate the biological *immune-response*. This has been discussed in detail in chapter 2 where the process of *clonal selection* [Burnet, 1959] was described. Activated B-Cells (i.e. those which best recognise an antigen) proliferate, growing into a clone of cells. As clones grow, the immune system turns on a mutation mechanism that generates mutations in the genes that code for the antibody. These point mutations occur at very high frequency. This process (known as somatic mutation) when coupled with selection, results in B-Cells that have very high affinity matches with antigen. This process is directly modelled in the response mechanism, which is outlined in figure 3.4.

3.6 Experimental Approach

$SCHED1-IS$ is heavily based on the model proposed by [Hightower et al., 1995, Perelson et al., 1996]. As all previous experiments with this model have been conducted using binary match-functions based on the simple binary universes first suggested by [Farmer et al., 1986] we first needed to verify that the proposed model using integer values for genes and a match-function based on evaluating the schedule produced by combining antigen and antibody actually underwent a process of evolution; i.e. the overall fitness of the immune-systems evolved after 200 generations was greater than those of the random initial population. We also wished to confirm the three findings of the Hightower work, namely;

1. The greater the antigen expression rate, the faster the learning
2. The larger the expressed antibody repertoire, the faster the learning and the higher final fitness of the evolved population
3. Somatic mutation accelerates evolution and illustrates the Baldwin effect.

As a result of attempting to confirm these findings, we would also be able to deduce sensible values for the system parameters, particularly N and K.

However, even if we show that evolution does take place in the proposed system, the actual values of fitness achieved following evolution do not give *any* indication as to whether or not the system has any practical value. This can only be gauged by inducing an immune response from the evolved system, and testing whether or not schedules can be produced which provide satisfactory solutions to scenarios that were defined in the world in which the libraries evolved, and also in response to completely new scenarios. Therefore, although a brief series of experiments is performed in order to confirm that the $SCHED1-IS$ system can evolve, the majority of this work is directed towards evaluating the immune systems produced, in order to quantify how they perform in a scheduling environment.

The data that was used is now described in section 3.6.1, followed by a description of parameters that were common to all experiments performed.

3.6.1 Experimental Data

Antigen universes, (AUs), were generated based on a set of benchmark scheduling problems given by Morton&Pentico in [Morton and Pentico, 1993]. These problems have been commonly used in a large number of scheduling studies. Results are reported here for the problem known as jb11.ss, which was selected as being typical of a medium-sized problem from this set. This problem contains 15 jobs, to be processed on 5 machines, and is known to have an optimal solution where no job arrives late. Each AU generated contained 10 antigens — an antigen was generated by mutating the original arrival date for each job with probability p_u to another random date, in the range (0,300), subject to the condition that the new arrival date was at least pt days before the due-date of the job, where pt was the minimum processing time required to complete the job. (Note that this method does not guarantee that the resulting conditions can lead to an optimum schedule where no job is tardy.)

3.6.2 Common parameters

In all experiments, a population of 100 random individuals was generated, with each individual characterised by $(l = 5, c = 5)$ and therefore $s = 15$ (as the total length of an antibody produced from an individual must equal the number of operations, 75). Thus, a total of $c^l = 3125$ antibodies can be formulated from a single immune system. These values were chosen after considerable experimentation with combinations of c and l. The value of c can be increased independently of l and s, however, clearly there is a trade-off between the amount of diversity that can be achieved within the system, and the amount of time required to evolve the system, given that selection pressure acts only on the phenotype. A similar trade-off exists in balancing l and s — the length s of the segment represents a common sequence of instructions for building a schedule. The likelihood of finding common sequences decreases as s increases, but the size of the search space increases exponentially as s decreases (and therefore l increases). The values of c, l, s reported here appeared to be a satisfactory compromise that allowed evolution to take place over a tractable amount of time, yet still produce satisfactory results.

All genes were randomly initialised with values in the range (0-14) as there were 15 jobs. The genetic algorithm used a generational reproduction strategy, with recombination performed by tournament selection of size 5, and uniform crossover. Each experiment was run for 200 generations, and repeated 10 times. All antigen universes contained 10 antigens and unless stated otherwise were generated by setting $p_u = 0.2$. Setting the tournament size to 5 exerts a high selection pressure, however given the nature of the fitness function, this was found to produce better results than when using a smaller tournament size.

3.6.3 Verification of the Hightower Model

Figure 3.5 shows the results of an initial series of experiments that were performed to verify that our model did indeed exhibit the characteristics expected (see section 3.6). All experiments were repeated 10 times, and the best fitness found averaged. Firstly, the antigen exposure rate was held constant at $K = 2$ (20% of the potential pathogen repertoire) whilst the antibody expression rate was varied. Next, the antibody expression rate was fixed at $N = 15$ (0.005% of c^l), and the antigen exposure rate varied between 20% and 100% of the potential number of antigens. Finally, some experiments were performed to confirm that somatic mutation could accelerate the evolution process. The optimal number of genes to undergo somatic mutation was varied and the resulting fitness of the immune systems measured, as well as the fitness trajectories over the course of the evolution when somatic mutation was applied.

Figure 3.5(a) shows that performance clearly increases as N is increased, as would be expected with greater sampling of the genetic material available in the genotype. Even when only 0.001% of the potential antibody repertoire is expressed, some evolution of the genetic material takes place, though somewhat slowly. When 0.05% of the repertoire is expressed, evolution is rapid, and approaches the maximum possible fitness of 1.0. There is clearly a trade-off however, between the time required to perform the evolution at large N against the fitness of the final system.

On the other hand, figure 3.5(b) shows that the average fitness of the evolved libraries decreases as the antigen exposure rate K is increased. This result conflicts directly with the work by Hightower *et al.*, [Hightower et al., 1995], in which they find

that *binary* immune systems evolve faster and end up with higher fitness values as the antigen exposure rate is increased. A possible explanation for this is that the 'optimal' schedules corresponding to each of the antigens in the universe are so diverse that especially at low N the expressed antibody repertoire is too general to provide reasonable solutions. Furthermore, in a binary system, there is a higher probability of a completely random antibody matching any given antigen than in the integer-based system modelled here.

Figure 3.5 (c) replicates the findings of [Perelson et al., 1996], in that increasing the number of points in the genome which are mutated improves fitness sharply over a narrow range of values of M, as a result of the Baldwin effect. Figure 3.5 (d), which compares the evolving fitness of the immune system with increasing generations for experiments that do and do not include somatic mutation, shows that including somatic mutation results in a more rapid evolution, and results in a higher overall fitness, again, as predicted in [Perelson et al., 1996].

Therefore, we conclude that the integer-based model with the interaction formed by the Ag-Ab complex describing a schedule is suitable for evolving immune libraries from which schedules can be generated. Best results will be achieved using low values of K, the antigen exposure rate, and high values of N, the antibody expression rate, and an element of somatic mutation. Given the time constraints introduced when using large values of N, we elected to set $N = 0.005$ in remaining experiments described, as evolution still occurs even at this level. Unless otherwise stated, the antigen exposure rate was set to 4.

3.7 Evaluation of the Immune Response

The experimental procedure adopted for evaluation of the libraries evolved using the genetic algorithm consisted of three distinct phases:

1. Measure the quality of the schedules produced from evolved libraries in response to scenarios described in the AU the library was evolved in.

2. Measure the quality of the schedules produced from evolved libraries in response to previously unseen scenarios, i.e. generate a set of new antigen universes

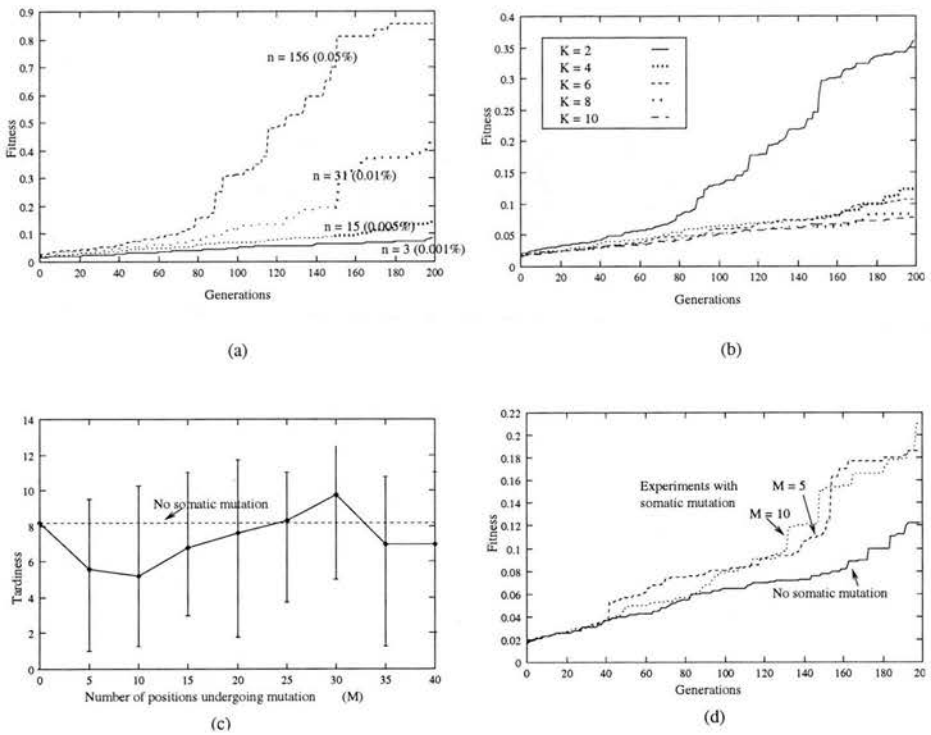

Figure 3.5:

(a) Fitness trajectory of GA experiments in which the antigen exposure rate is held constant at $K = 4$, and the antibody expression rate varied.

(b) Fitness trajectory of GA experiments in which the antibody expression rate is held constant at N=15, and the antigen exposure rate K varied between 2 and 10.

(c) Effect on average tardiness of schedules of increasing the number of genes M that undergo somatic mutation at each generation of the GA. The antibody expression rate was fixed at $N = 15$ and the antigen exposure rate $K = 4$.

(d) Fitness trajectory of GA experiments in which the somatic mutation rate is varied. The graphs show that applying somatic mutation accelerates evolution. Parameters were set as in (c).

3. Define a measure of *robustness* of a schedule, and evaluate evolved schedules against this measure.

Schedules were produced from evolved libraries using the algorithm outlined in figure 3.4. Preliminary experiments established suitable values for the parameters of this algorithm. The quality of the schedules generated was compared against a schedule evolved using the specialised scheduling GA proposed by Fang in [Fang et al., 1993]. The fitness of the respective schedules was compared, as well as the actual schedules themselves. In order to provide a fair comparison, Fang's genetic algorithm is run for the same number of generations and using the same parameters and operators as used in evolving the immune libraries. However, note that a single generation of SCHED1-IS involves evaluating $(K * N * p)$ schedules, where K is the number of antigens the system is exposed to, N is the number of antibodies expressed and p is the size of the population, whereas Fang's GA requires evaluating only p schedules per generation. However, Fang's GA must be run once for *every* scheduling problem to be solved, therefore the total number of generations required to produce solutions to *all* problems must be multiplied by N_a, the number of antigens in the universe. One run of $SCHED1 - IS$ on the other hand, solves all problems at the same time, and as typically, $K << N_a$ (in the experiments performed $K = 4$ and $N_a = 10$), the effort may not differ significantly between the two methods. Furthermore, even if more effort is initially required to evolve an immune library, once it has evolved, it then contains the material required to produce new schedules quickly and efficiently so the initial effort is not wasted. Using Fang's GA to construct a new schedule requires the whole process to start from scratch, and therefore a further $p * g$ schedule evaluations, where g is the number of generations.

As stated in the introductory section of this chapter, one of the aims of this work is to produce schedules that are *robust*, i.e. can absorb changes within the operating environment. This can be interpreted as producing schedules which cover more than one contingency — if this is the case, a schedule does not necessarily have to be changed if the conditions under which it was produced change. Thus, even if the quality of schedules produced from the immune libraries is favourable compared to those produced by the Fang algorithm, if the schedules are significantly different from each other they

machine	job order				
A	1	3	2	5	4
B	2	5	4	3	1
C	1	2	3	4	5
D	5	4	3	2	1
E	1	3	2	4	5

machine	job order				
A	3	1	2	5	4
B	2	5	4	3	1
C	1	2	3	4	5
D	5	4	2	3	1
E	1	3	2	4	5

Figure 3.6: Comparison of two schedules — cells in which the two schedules differ are shaded

may cause much disruption if they were to be put into practice.

Therefore, we introduce a measure of similarity of two schedules. Given a schedule which describes the operation of J jobs on M machines, then we can write a schedule S as a matrix of m rows and j columns. Each of the m rows represents a machine, and the row indicates the order in which jobs are processed on that machine. Thus $S_{(i,j)}$ represents the ith job to occur on the jth machine. An example is shown in figure 3.6.

We suggest that two schedules can be considered similar if jobs are processed on a machine in the *same* order in each schedule, regardless of the time that the jobs start or finish. Thus, if we compare two matrices and count the number of cells in which the matrices differ, we have a quantatitive means of comparing the similarity of two schedules S and S'. This robustness measure R is defined in equation 3.2. In the problem described, we have 15 jobs, each of which is to be scheduled on 5 machines. Given two permutations of $1...n$ numbers, then it can be shown (appendix A) that the number of expected coincidences between them is just 1. Thus, in a scheduling problem with j random jobs on m machines, a random schedule contains j permutations of $1...m$, and so the expected number of places of similarity between two schedules is simply j with variance j and standard deviation $\sqrt{j}$.

$$R = \sum_{j,m} \begin{cases} 0 & if \quad S_{(j,m)} = S'_{(j,m)} \\ 1 & otherwise \end{cases} \tag{3.2}$$

3.7.1 Selecting the Clone rate, Antibody Expression rate and Mutation rate

This section describes a series of experiments which investigated the effect and importance of the choice of values for N, the number of antibodies expressed, C, the number of clones produced, and p_m, the mutation rate, in producing a response. A set of immune libraries evolved using $N = 0.005\%$ and $K = 4$ was used as the test immune system, and a response was generated 100 times to each of the 10 antigens present in the AU that the immune libraries were evolved against. A response was generated 100 times against each antigen using a set of parameters (N, C, p_m), and the fitness of the resulting schedule measured. All results were averaged over the 100 responses. The patterns that emerged were identical for all antigens — results are shown for the response against a single antigen in figure 3.7. The fittest schedules are obtained at low mutation rates, and at high values of both N and C.

As a result of this, in all further experiments, the values of N and C were each set to 1000, and p_m to 0.2. Given that we used 5 libraries, with component size $s = 15$, this is equivalent to expressing approximately 32% of the potential repertoire.

3.7.2 Comparison of tardiness of schedules produced from $SCHED1-IS$ to those produced by Fang GA

For each antigen-exposure rate, K, tested in section 3.6.3, the best set of immune-libraries produced in the 10 experiments was used to produce schedules for each of the 10 antigens in the original antigen universe ($p = 0.2$) in which the libraries were evolved, and then for three further antigen universes generated using $p_u = 0.1$, $p_u = 0.3$ and $p_u = 0.5$. This involved applying algorithm 3.4 100 times, using the parameters determined above, and comparing the average tardiness of the *best* schedules found to the tardiness of the corresponding schedule produced via the Fang algorithm. Table 3.1 shows the percentage of the 10 antigens in each universe for which the *best* schedule found using the evolved immune-libraries was superior to that found by Fang. The figures in brackets give the corresponding percentage values for the *average* tardiness.

Although there is no clear-cut trend with increasing K, for most values of p_u there

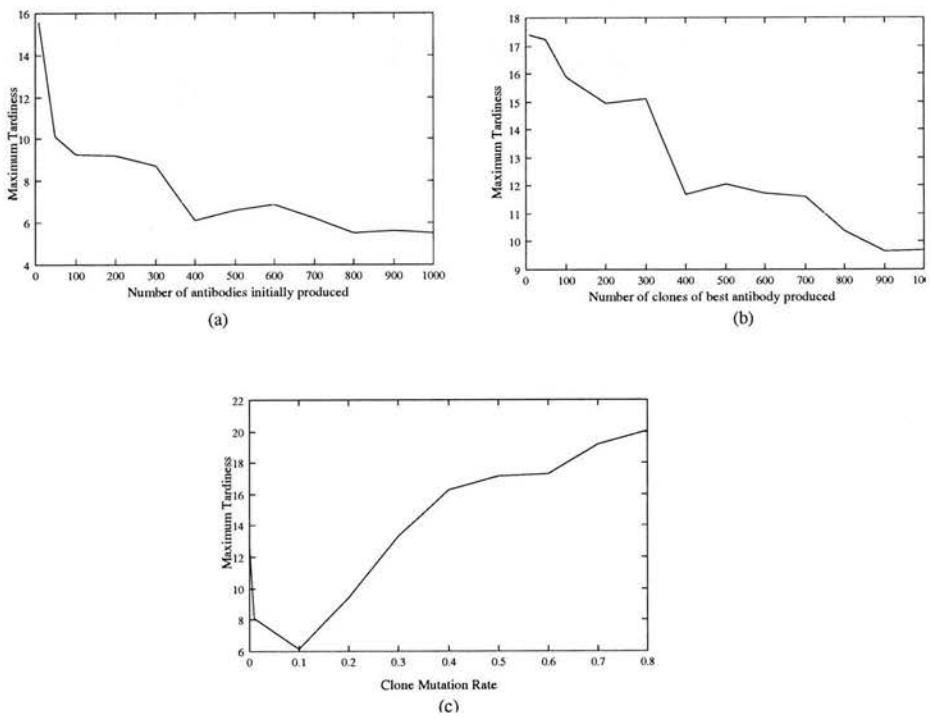

Figure 3.7:

(a) Effect of varying the number of initial antibodies generated N during the immune response on the average tardiness of schedules. C is fixed at 1000, and p_m at 0.2

(b) Effect of varying the clone rate C during the immune response on the average tardiness of schedules produced. N is fixed at 100, and p_m at 0.2

(c) Effect of varying the mutation rate p_m during the immune response on the average tardiness of schedules produced. N is fixed at 100, and C at 1000

Antigen Universe	Antigen Exposure During Evolution of *AIS*				
p_u	2	4	6	8	10
0.1	20 (50)	80 (90)	20 (50)	70 (80)	60 (80)
0.2	10 (20)	30 (40)	20 (30)	40 (40)	60 (60)
0.3	0 (20)	50 (50)	30 (30)	60 (60)	40 (50)
0.5	0 (0)	40 (30)	10 (0)	40 (20)	0 (0)

Table 3.1: Percentage of test-cases where *best* and *(average)* tardiness of *AIS* schedule was equal to or less than result found by Fang

is a general tendency for the schedules produced via the immune-libraries to increase in quality as K increases, despite the fact that the overall evolution of those libraries is slower (see section 3.6.3). Examination of any trends occurring as the diversity of the antigen universe increases (i.e. p_u) shows that the ability of the immune-libraries to match the results produced by Fang decreases as p_u increases. Even at $p_u = 0.5$ however, using the *AIS* from $K = 8$ we are able to match the Fang results in 40% of cases, which is somewhat surprising, considering the wide diversity of arrival-dates amongst the antigens in this universe. In general, libraries evolved at a low antigen exposure (K=2) perform badly. Libraries evolved when $K = 4$ and $K = 8$ generally performed well across all universes.

It is worth noting that in order to produce a schedule using Fang's specialised GA requires 20,000 evaluations of schedules (i.e. 100 individuals over 200 generations). On the other hand, producing schedules from the *evolved* immune-libraries requires at most 2000 evaluations:- evaluation of 1000 initial antibodies, followed by evaluation of 1000 clones. Furthermore, figure 3.7 (a) suggests that the number of antibodies initially produced could also be significantly reduced, perhaps to 400, as increasing the number beyond this point does not produce a corresponding increase in performance.

	p_u			
	0.1	0.2	0.3	0.5
AIS	30.11	46.48	41.47	44.31
Fang	43.86	44.58	45.49	46.20

Table 3.2: Average robustness R of schedules in different antigen universes

3.7.3 Robustness of Schedules

In order to analyse the *robustness* of schedules produced from the immune-libraries (see section3.7), a schedule was generated from an immune-library evolved using $(N = 0.005, K = 4)$ for each of the 10 antigens in each universe characterised by $p_u \in 0.1, 0.2, 0.3, 0.5$. A pairwise comparison of each of the 10 schedules in each universe was performed, using R as the measure of comparison. This was repeated comparing schedules generated using the Fang algorithm. The average value of R in each case is given in table 3.2.

Apart from the anomalous case of $p_u = 0.2$, we see an improvement gained by producing schedules from the immune-libraries, in that the average value of R is lower and therefore the schedules more similar. In section 3.7 it was noted that the expected value of R is j — note that in all cases, the value of R obtained using the immune-libraries is greater than $2j!$. Moreover, Chebyshev's inequality suggests that the probability of getting 40 or more coincidences between two 15-job schedules is $\leq 15/(40-15)^2 = 2.4\%$, and therefore the probability of obtaining these results by chance is very low. The difference in R between the library generated schedules and the Fang schedules decreases as p_u increases, and is also very high for high values of p_u. This is probably due to the increased diversity in antigens that is obtained by mutating each arrival-date with high probability, and therefore the resulting low probability of producing a single schedule that can cover all cases effectively. (Obviously a *single* schedule could be produced to cover all contingencies by only considering the *latest* arrival date for every job across the whole antigen universe — this would result in a schedule with large amounts of idle time however, and possibly several very late jobs.)

3.8 Summary of Utility of $SCHED1-IS$

The experiments just described show some promise as a first attempt at designing a scheduling system based on immunological principles. In particular, they go some way towards meeting the goals originally outlined, i.e. that the system should be able to rapidly and efficiently produce schedules that are satisfactory in quality *and* robust to changes in the environment. This is confirmed by the experiments in sections 3.7.2 and 3.7.3. Aside from the scheduling perspective, it also lends further weight to the claims made originally by Hightower *et. al.* that selection operating at the organismic level can provide the selection pressure needed to generate and maintain diversity in an organisms gene-libraries, as our integer-based model is a more generic representation of the libraries themselves and the match-function more complex; remember that genes in the biological immune system are composed from *four* possible bases — A,D,G T — rather than the *two* modelled in Farmer's original binary artificial immune system, and the interaction or binding affinity between antigen and antibody is clearly more complex than simply summing numbers of complementary bits.

However, so far only a cursory investigation has been performed of the use of such a system for scheduling, using restricted and small antigen universes, in which only one type of contingency (inconsistency in arrival-dates) has been examined. With hindsight, when taking a more global view of the overall picture, the model seems to embody two obvious flaws:

Representation — the information contained in a single library component changes its meaning according to the components preceding it in an antibody. For example, a '1' at some position i in the list of instructions for building a schedule means schedule the next operation of job 1. However, the exact identity of this operation is dependent on how many operations of job 1 have already been scheduled in the previous $(l-i)$ instructions. This is likely to be a consequence of using *any* indirect representation, however it is difficult to envisage how anything other than an indirect representation could be used in a library/component model as described, if it is essential that combining random components from libraries always leads to feasible antibodies.

The Immune Response — even if a suitable set of libraries of components can be produced, then the problem of how to retrieve antibodies from the immune system still exists. Selecting random components and combining them to produce antibodies results in an antibody search space of size c^l. There could be considerable variation in quality across this space, and hence it could be counterproductive to spend much effort searching it. In the experiments described, the size of the complete antibody search-space was 3125, (5^5), which is small compared to the number of schedule evaluations generally required by a GA to solve such a problem, which is often of the order of 20,000, see [Fang et al., 1993, Lin et al., 1997]. However, in large problems which require larger immune systems, the combinatorics may readily become intractable.

In light of these inadequacies, the entire motivation behind trying to produce immune-libraries that represented repositories of schedule fragments from which *robust* schedules could be produced was re-examined. Analysis of data supplied by real companies relating to their scheduling problems, for example [Hart et al., 1998, Marshalls Agriculture, 1998], revealed that similar scenarios often crop up over and over again, and as a result there are known methods for dealing with them. An experienced scheduler can quickly piece together new schedules using prior knowledge gained from past experiences. Therefore, the problem faced by a human scheduler is generally not how to produce a new schedule starting from scratch, or indeed how to originally design a schedule that is robust to a range of potential conditions, but how to select and draw on prior experiences to adapt an existing schedule to a new situation.

Therefore, we now propose that addressing the more fundamental question of how to evolve the contents of the actual immune libraries so that the libraries contain *robust* building blocks is inappropriate — a more pertinent question is how can we construct an immune system in which the library components represent a store of *prior experiences*. In theory, these experiences can be pieced together and perhaps undergo adaptation in some manner in order to quickly make new schedules. In fact, this can be considered directly analogous to the secondary response in the biological immune system in which a persistent sub-population of memory cells retains information about previous antigenic attacks — these cells are restimulated either by attack from the

original antigen, [Jerne, 1973, Tew and Mandel, 1979] or by attack from a related environmental antigen [Matzinger, 1994a].

3.9 $SCHED2-IS$ — Storing Past Experiences in an Immune Library

Thus a second model is now proposed, based on immunological principles, and by which a scheduling system might be implemented. This model is referred to as $SCHED2-IS$. The model is based on the conjecture that although a variety of conditions may arise that all require a new schedule to be implemented, the conditions are often predictable and have associated predictable (partial) solutions. Therefore, the most relevant of the immunological principles to capture within the model appears to be that of *memory*, which allows the prior experiences to be maintained. By also modelling some of the *diversity generation* processes observed in the biological system, it should be possible to enable those cells to adapt in order to recognise new invaders more specifically. We also speculate that although many slight deviations from the norm arise in a scheduling environment, it is *rare* to have to radically reschedule. Therefore although the biological immune system is able to respond to entirely new situations owing to its astonishingly effective diversity generating mechanisms, it is unnecessary to rely on these mechanisms in *most* of the day to day situations that occur.

We propose that when scheduling is performed by an experienced *human*, then he or she relies on using historical information in order to construct new schedules — this information may relate to specific past events, or to long or short sequences of past events, and is commonly described as 'intuition' — something that is difficult if not impossible to capture in a computer model. For example, it is not difficult to imagine some or all of the following thought processes occurring to our experienced scheduler when faced with a scheduling scenario:

- *Job A and Job B can usually be performed in parallel*
- *Operations a,b,c tend to occur in a group in many schedules, but in different*

permutations

- *When machine X breaks down, often task Z can be performed while waiting*
- *Job C is often late arriving, so process Job D instead*

These processes are captured to some extent in the historical schedules themselves; For example, if we examine the order in which jobs are processed on a specific machine, or the order in which jobs are selected from the shop-floor for processing, then it is common to observe *patterns* occurring across subsets of these processes. Thus, if a set of common patterns or parts of schedules could be built up using the knowledge encapsulated in past schedules, then these patterns can be used as *building blocks* when constructing a new schedule. The simple idea is that building blocks formulated as a result of past experiences encapsulate past learning and knowledge, and therefore should be an efficient and rapid way of forming a new schedule. This type of approach to scheduling also seems more realistic than the typical GA approach which starts from a random starting point and searches for a new schedule, using no knowledge of past behaviour.

Thus, in contrast to $SCHED1-IS$ which adopts a *bottom-up* approach to producing schedules, we are now proposing a *top-down* approach. Figure 3.8 shows the major differences in the approach taken in each case. In $SCHED1-IS$, the system starts with a set of potential scheduling conditions, and generates an evolved immune system consisting of libraries of partial instructions for creating schedules. In contrast, $SCHED2-IS$ starts with historical information describing actual past schedules; a set of building blocks is derived from these schedules, which can then be recombined to produce new schedules. Thus, although both models ultimately result in a step which builds new schedules from smaller building blocks, the preliminary steps taken to generate these building blocks are very different.

3.10 $SCHED2-IS$ - Description of Model

This section describes the new model, $SCHED2-IS$, which adopts the top-down approach outlined in figure 3.8. The model must satisfy two aims; firstly, derive a sensible

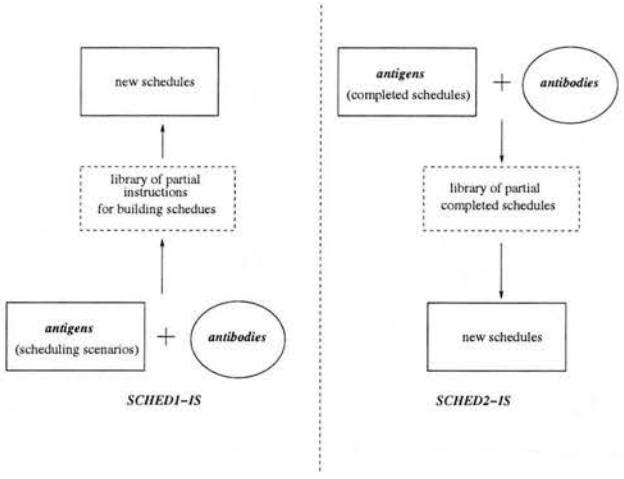

Figure 3.8: Comparison of the two proposed models for a scheduling immune system, $SCHED1-IS$ and $SCHED2-IS$

set of building blocks, and secondly, be able to recombine those building blocks into new schedules. What constitutes a useful set of building blocks? In order to answer this question, we must first consider what exactly it is that those building blocks represent. It has already been mentioned that there are a number of obvious processes occurring whilst scheduling in which patterns can be observed. The most obvious, and that chosen for use in this study is the pattern of job-sequences observed on individual machines in historical schedules. The approach adopted however does not preclude other types of pattern; for example distribution of idle times on machines, or the order in which jobs are placed in the schedule. When considering job-sequences, although it is likely that some subsets of historical schedules will contain common job-sequences on certain machines, it is unlikely a common sequence of jobs will be observed in *all* schedules. Thus, ideally we wish the building block set to contain as many common sequences as possible, but at the same time, contain specialist sequences that are only applicable to certain unique situations that may arise again in the future. The next section describes one method by which a building block set can be generated. Section 3.10.2 then describes the approach taken in $SCHED2-IS$ to recombining these blocks into schedules. This is followed by some experimental results in which the performance of the model is thoroughly analysed.

3.10.1 Deriving the building blocks

As stated above, the proposed model must incorporate a method of producing a *set* of diverse building blocks, which includes both *specialist* blocks — those that represent a unique pattern occurring in a schedule, and *generalist* blocks — those which represent a common pattern occurring in many schedules. The genetic algorithm has been advocated as a method of searching for a population of structures that jointly perform a computational task[1], for example in a learning classifier system [Holland et al., 1986]. In such problems, the GA must search for a set of individuals which are specialised to various tasks or niches; collectively the individuals provide a complete solution to the problem being solved. Depending on the number of individuals in the populations, and the number of niches in the problem, some of the evolved individuals must generalise in order to cover more than one peak, whilst others can remain more specialist. In general, the difficulty with using a GA to solve such a problem is how to retain sufficient diversity within the population in order to cover all the niches. Also, GA approaches tend to suffer from the drawback that it is impossible to track more niches than members of the population. Many methods have been suggested, the details of which are beyond the scope of this thesis. These include crowding [DeJong, 1975], assortative mating algorithms, [Booker, 1985], and dividing the population into sub-populations, [Whitely and Starkweather, 1990]. Three methods seemed worthy of further investigation:

3.10.1.1 Fitness Sharing

[Deb and Goldberg, 1989, Goldberg and Richardson, 1987] introduced the notion of fitness sharing. The idea behind this method is that diversity is maintained in a population by punishing individuals that are similar to other individuals within the population, and is rooted in ecological ideas that in an environment consisting of multiple niches, there are finite resources available for each niche. The method appears to work well in some situations but has significant limitations ([Smith et al., 1993]):

- It is necessary to know *a priori* how many niches there are in the environment

[1] although the majority of GA applications tend to be directed towards evolving a *single* population member that specifies a single optimised solution.

- It is dependent on a uniform distribution of peaks in the search space
- It requires a comparison of every population member to every other population member in each generation, i.e. N^2 comparisons for a population of size N, therefore is time-consuming.

For these reasons, the approach did not seem suitable to the task of discovering schedule building blocks; we do not know how many blocks (i.e. niches) are required, and they are unlikely to be evenly distributed across the search space. Therefore this approach was rejected for inclusion in $SCHED2-IS$.

3.10.1.2 'Pitt Approach'

In this approach to classifier systems, described in [DeJong, 9898], a population of rules is concatenated into a single individual which is then manipulated by a GA. This allows diversity to be maintained within each rule set but is inherently inefficient as it manipulates whole rule-sets, rather than populations of rules. Furthermore, this method also requires that the number of rules (or building blocks in the case of $SCHED2-IS$) is pre-judged in order to form an individual chromosome, although a reasonable 'guess' will suffice. Therefore, it was also rejected as the engine for discovering building-blocks in $SCHED2-IS$.

3.10.1.3 An Immune System Model

[Smith et al., 1993] proposed a theoretical model of an immune system which can be used to evolve a set of *antibodies* that recognise a range of diverse, binary antigen strings. This work (verified experimentally in [Forrest et al., 1993]) showed that an immune system model could both detect common patterns (schemas in the binary case) in a noisy environment and also maintain diversity in that many types of antibody evolved in *niches*, each niche responsible for recognising a particular antigen. Its success lies in the novel fitness scheme introduced, referred to as the diversity algorithm. (This work has previously been described in detail in chapter 2.) This model has three appealing features as far as our scheduling system is concerned.

1. It can solve problems in which the niches are not equally spaced, which is evidently the case in the scheduling domain described.

2. It does not require explicit construction of a sharing function, and therefore does not rely on any *a priori* knowledge of the number of niches.

3. it is possible to control the evolution of the antibodies representing the niches to be either *specialist* (i.e the antibody only recognised a single specific antigen), or *generalist* (i.e the antibody recognised a wide range of antigens) by varying the parameters of the fitness function proposed.

This model therefore seems ideal for our purposes. However, before adopting it wholesale, a more careful consideration of the characteristics of the building blocks we wish to evolve is required. Each antibody or building block will 'match' a subset of the historical schedules. The subsets may intersect or be disjoint. Figure 3.9 shows a simplified view of the situation; a population of antibodies is depicted, and the diagram shows how they match four antigens, labelled A,B,C and D. The diagram illustrates the point that we are interested in two particular attributes of the antibody population:

1. *Overlap* — i.e. how many *antigens* a given *antibody* matches. Overlap can be considered as a measure of how common the pattern represented by the antibody is in the schedule set.

2. *Redundancy* — i..e. the number of different *antibodies* matching an *antigen*. A schedule may contain several sequences, each of which is common to different subset of the remaining schedules, therefore more than one antibody may match a single antigen.

Although the work of [Smith et al., 1993] implied that their diversity algorithm would allow these properties to be controlled, their algorithm was re-implemented exactly as described in [Forrest et al., 1993] and a number of tests were carried out using binary antigen and antibody strings in order to quantify exactly the amount of *overlap* and *redundancy* that would occur using large antibody populations in antigen universes which varied in size. The aim was to give more insight into whether the algorithm

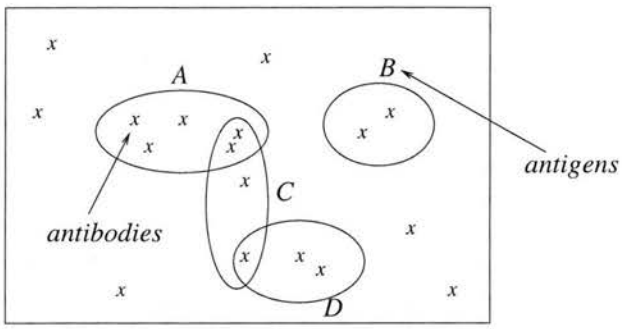

Figure 3.9: A Population of Antibodies, showing how they match 4 antigens A,B,C,D

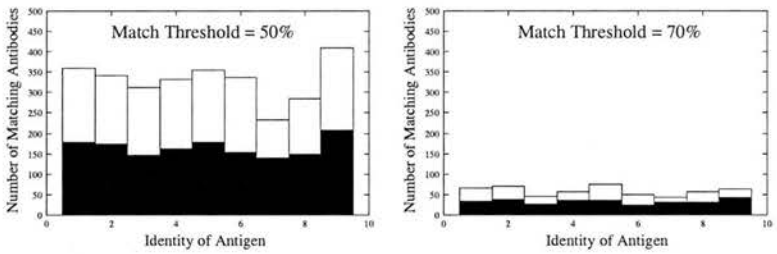

Figure 3.10: Graphs show the effect of redundancy when using a GA to evolve antibodies matching binary antigens. The shaded portion of the graph represents the fraction of *unique* antibodies found

could be scaled to reproduce these characteristics in a more diverse environment. A population of 500 antibodies was chosen, each of length 64 bits. The antigens used are given in table 3.3, based on those described in [Forrest et al., 1993]. The parameters used for the GA and experiments are identical to that given in [Forrest et al., 1993]. According to Forrest, a match is said to occur if at least m percent of the bits in the antigen and antibody match, where m is known as the match-threshold. We tested two values of match-threshold — a low value of 50%, and a higher one of 70%.

Figure 3.10 shows the number of antibodies matching each individual antigen in the case where there were 9 antigens for both values of m. The shaded fraction of each bar indicates how many of those antibodies were *unique*. The figure clearly shows that at high match-thresholds, small niches of antibodies occur, but that there is a high proportion of diversity within the niche, i.e. many different antibodies recognise the same antigen. This is promising for the proposed scheduling system.

Antigen ID	Antigen Definition
1	00000000....00000000
2	11111111....11111111
3	10101010....10101010
4	01010101....01010101
5	11001100....11001100
6	00110011....00110011
7	11101110....11101110
8	00010001....00010001
9	11110000....11110000

Table 3.3: Antigens used to quantify the *overlap* and *redundancy* occurring in antibody populations evolved using the diversity algorithm proposed in [Forrest et al., 1993]

Figure 3.11 shows the amount of overlap in the same populations, by measuring how many of the antibodies in the final population *that match at least one antigen* match more than one antigen. Clearly, if the match-threshold is high, there is very little overlap between antibodies. This is not particularly convenient for our proposed scheduling system, in which we wish to determine building blocks that are common to several schedules, i.e. niches.

Therefore the results of these preliminary investigations pointed to the fact that Smith's diversity algorithm would need some modifications before it was suitable for producing building blocks for a scheduling system, in order to encourage more overlap as well as maintaining some level of redundancy and niche formation. The implementation of the modified algorithm is described in full in section 3.11.

3.10.2 Recombining Building Blocks To Form A Schedule

The previous section has described how an immune system model might be used to evolve a set of short antibodies (job-sequences) that at least partially match a set of antigens (schedules). The biological immune system responds to immunological challenges by manufacturing B-Cells using information encoded in its germline DNA. If

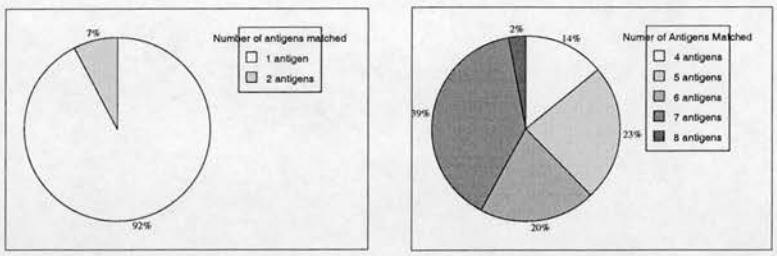

Figure 3.11: Measuring overlap: The graph shows the percentage of antibodies in the final population of matching antibodies that match more than one antigen. The left hand figure shows the result for a match-threshold 70%, the right hand figure for match-threshold 50%.

we consider our evolved antibody population to represent this germline DNA, then in a similar manner we wish to construct complete schedules using this evolved information.

The astonishing diversity that can be produced from a relatively small genetic base in the biological immune system has already been alluded to earlier in this thesis (remember that the human immune system can produce more antibodies than there are genes in our genome!). Many mechanisms have been proposed by which this may occur. These include, for example, combinatorial rearrangement of entries from multiple libraries [Tonegawa, 1983], junctional diversity [Gilfillan et al., 1993], and somatic hypermutation and/or gene conversion [Weigert et al., 1970]. The exact nature of these mechanisms is complex, however they are shown schematically in figure 3.12. Figure 3.12(a) shows the most straightforward diversity recombination mechanism — multiple segments of DNA are randomly recombined to form new antibodies. Figure 3.12(b) shows an example of somatic recombination — in this mechanism, imprecise joining of segments occurs during recombination. Part (c) of this figure illustrates nucleotide addition — when two segments are cut and then joined, extra nucleotides are sometimes inserted between the joins, leading to further diversification. $SCHED2-IS$ implements versions of these three mechanisms in order to recombine antibodies in order to produce complete schedules. It is proposed that this is an efficient method of producing complete schedules, taking account of historical experiences. For example,

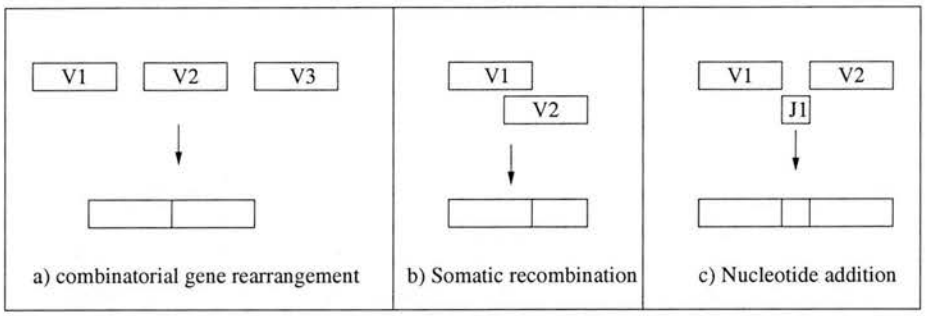

Figure 3.12: The figure shows three mechanisms by which the biological immune system generates diversity by combining *variable* gene regions.
(a) Combinatorial gene rearrangement: a large number of genes in the germline can randomly recombine to form a new protein
(b) Somatic recombination: imperfect recombination of genes sometimes results in new proteins that are shorter than the original recombining segments
(c) Nucleotide addition: extra nucleotides encoding for additional amino acids are sometimes inserted between two joining segments

assume that the germ-line contains n DNA chunks or antibodies representing a partial sequence of jobs, each of length l, and that a *minimum* of s chunks are required to construct a schedule. At least $C = \binom{n}{s}$ possible combinations of these antibodies can be produced. Now, for a scenario in which j jobs need to be scheduled on a single machine, then examining all possible sequences of these jobs would result in $j!$ possible schedules. However, we postulate that C_v, the number of *valid schedules* in C is $\ll j!$ for two reasons:

1. Many of the theoretical $\binom{n}{s}$ schedules will contain multiple instances of jobs or missing jobs, and hence the schedules are illegal and can be discarded.

2. The n partial schedules encapsulate prior knowledge, and hence are guaranteed to be suitable subsequences, i.e to be the most promising of the $l!$ possible subsequences.

Therefore, if a suitable method can be found of combining the antibodies evolved using $SCHED2-IS$, then the complete system should represent an efficient method of

producing new schedules to react to changing circumstances. The implementation of these recombination methods is described in section 3.11.

3.11 Implementation of $SCHED2-IS$

This section now describes the manner in which the model that has just been outlined was actually implemented. It is followed by a description of experiments that were performed in order to analyse the model's performance.

3.11.1 Antigen Representation

As stated in section 3.10, the antibodies produced by $SCHED2-IS$ will represent patterns occurring in job-sequences on each machine in a set of historical schedules. Therefore an *antigen* simply consists of an integer-string defining the sequence of jobs observed on a *single* machine. For a problem in which jobs are to be scheduled on m machines, then m antigens are generated. If there are j jobs to be scheduled, the length of each antigen is also j.

3.11.2 Antibody Representation

An antibody is represented by a sequence of integers, of length l, where $l < j$. l is chosen to be significantly less than j as it is expected that the common patterns will consist of short sequences of jobs. A wild-card gene was also introduced which could match all possible jobs; the importance of including a gene which can match other genes in a non-binary system has been discussed in [Cooke and Hunt, 1995, Hunt and Cooke, 1996], who proposed an artificial immune system model to recognise promoter sequences in real DNA sequences. They found that the task could not be accomplished without introducing a wild-card that could match any of the bases A,C,T,G. For example, consider the following three job sequences:

```
1 3 2 5 9 8 7 4 6
1 3 2 9 8 5 7 4 6
1 3 2 8 9 4 7 4 6
```

If the wild-card gene is allowed, they can be matched by a *single* antibody

```
1 3 2 * * * 7 4 6
```

A further advantage of this approach is that it is impossible to judge *a priori* the optimum value for l. If many of the common job-sequences are shorter than the chosen antibody length l, then a partially matching antibody containing a number of wild-cards can still have high fitness. An antibody is prevented from containing duplicate jobs, though it may contain multiple copies of the wild-card allele '*'.

3.11.3 The Matching Algorithm

In order to quantify the extent of a match between an antigen and antibody string, a *match function, M* must be defined, such that $M : Antigen \times Antibody \rightarrow \mathcal{R}$. Reviewing the immunological literature suggest many physiologically plausible match-rules, for example see [Perelson, 1989]. The simplest of these, adopted in [Forrest et al., 1993, Hightower et al., 1995] in binary immune studies is simply to sum the number of complementary bits between antigen and antibody. [Stadnyk, 1987] introduced a more complex function that computes the length of each complementary region and then combines them in a manner which rewards longer regions more highly than short ones. [Cooke and Hunt, 1995, Hunt et al., 1995] introduced a version of this match-rule shown in figure 3.13, which was used to quantify matches in non-binary data (DNA promoter sequences and the publically available cabata case base describing various features associated with holidays such as type, cost, duration, location etc.). This function is weighted in favour of contiguous matching regions. Furthermore, [Farmer et al., 1986] and also [Hunt and Cooke, 1996] suggest using a function in which a match is allowed to start at any point on the antigen. The function suggested by Farmer sums all possible matches found in this way, with the rationale that molecules may be able to interact in more than one way, and thus react more strongly because they spend more time together than molecules that can only interact in one alignment. Hunt *et. al.* take the alternative view that the match-score is equivalent to the maximum score found when considering all possible alignments.

We choose to use a simple match-function which simply counts the number of

1. c = number of fields that match between the antibody and antigen
2. For each region consisting of 2 or more contiguous matches, record their length l_i
3. $M = c + \sum_i 2^{l_i}$

Figure 3.13: Match function introduced by [Cooke and Hunt, 1995]

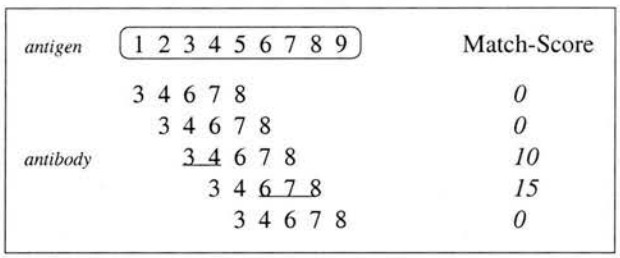

Figure 3.14: Possible Alignments of an antibody with an antigen, and the resulting match-score

matching genes between antigen and antibody, but also incorporates the suggestion of Farmer *et al.* and Cooke *et al.* that more than one possible alignment of antigen and antibody should be considered. The method is as follows:

An antibody is matched against an antigen by aligning the two strings. If the antibody is shorter than the antigen, then a match-score is calculated for every possible alignment position, where a possible alignment is any alignment in which every gene of the antibody is aligned with every gene of the antigen. This is illustrated in figure 3.14. The match-score is then calculated by counting the number of matches between antigen and antibody genes in the alignment. An exact match contributes a score of 5, whereas a wild card match contributes a score of 1. This prevents the evolution of all antibodies containing only wild-card genes — a similar method was used in [Cooke and Hunt, 1995]. All possible match-scores are calculated, and the maximum value found assigned to the antibody.

1. Select a sample of antigens of size τ at random and without replacement.
2. Select a sample of antibodies of size σ at random and without replacement.
3. Match each antibody in the sample against each antigen, summing the match-scores obtained for each antigen to give the total match-score for the antibody.
4. The antibody with the highest total match-score has this score added to its fitness. The fitness of all other antibodies in the sample remains unchanged.
5. Repeat the process for typically three times the number of antigens.

Figure 3.15: The modified fitness scheme used to assign fitness to antibodies in the population

3.11.4 An Emergent Fitness Sharing Function

The results of the preliminary experiments described in section 3.10.1.3 implied that the fitness scheme proposed by [Forrest et al., 1993] would need some modification in order to encourage the occurrence of overlapping antibodies in the system. The original scheme (similar to the "best-match" strategy used in classifier systems) selected a *single* antigen at random from the antigen universe, and matched it against a sample of the antibody population. In order to encourage overlap, this was replaced by a step that selected a *sample* of the antigen population of size τ, at random and without replacement, and matched them against a sample of the antibody population. The match-scores obtained between an antibody and *each* antigen in the sample are summed to give a total match-score for the antibody, and therefore reflects its cross-reactivity. The complete algorithm is given in figure 3.15.

3.11.5 The Genetic Algorithm

[Forrest et al., 1993] use a genetic algorithm based on GENESIS, [Grefenstette, 1984], in order to evolve the antibody population. As their work was concerned only with binary universes, it sufficed to use standard crossover and mutation operators such as two-point crossover. However, this is not applicable to a scheduling scenario in which we wish to maintain antibodies which *never* contain duplicate jobs. Three types of possible crossover operator were identified, including one novel one designed specifically for this application:

1. *Order-Based Crossover (OX)* — introduced by [Davis, 1985], this operator can be used if the parents are permutations of each other.

2. *2pt-Crossover* — if the parents do *not* have any genes in common, (excluding wild cards) and the parents differ outside of a randomly chosen cross-segment, then the standard two-point crossover operator can be applied.

3. *Overlap-Crossover* — this novel operator can be used if one parent *overlaps* the other, as shown in figure 3.16. In this case, parents are first aligned so that the matching regions line up. Reading from the left-most position, if only one parent has a gene at a position, then that gene is passed to the child. If both parents have a gene at a position, then a gene is selected randomly from either parent. This process is continued, reading from left to right until the child is of the required length. In figure 3.16, matching regions are underlined and shown in bold, and genes which can be chosen from either parent are shown in italics.

All three crossover operators are implemented, and used according to the relationship between the parents chosen for crossover, as described above.

A mutation operator is applied that randomly mutates each gene with probability $1/l$ to another randomly chosen allele, with the caveat that duplicate jobs cannot exist. Thus, the publically available GENESIS package was modified to accommodate these changes.

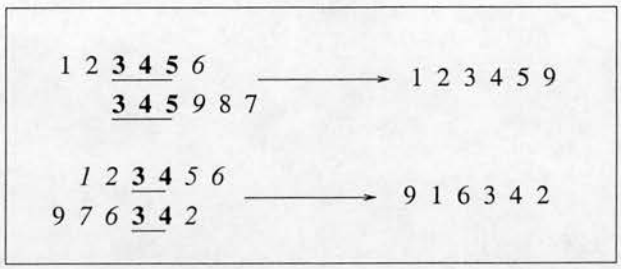

Figure 3.16: Overlap Crossover — the figure shows examples of two overlapping configurations of parent antibodies and the resulting child antibody in each case

3.11.6 Recombination Operators

Section 3.10.2 described three recombination mechanisms, each closely related to a feature observed in the natural immune system. The actual implementation of those mechanisms is now described. The output from the first phase of $SCHED2-IS$ is a set of antibodies, each of length l. The aim is to construct a schedule for each machine of length j, starting from a partial schedule of length l_p, where $l_p < j$. If $l_p = 0$ then the schedule is constructed from the beginning, otherwise the schedule is completed from the point at which the change triggering rescheduling occurred. The three mechanisms are as follows:

1. *Simple Recombination* — in this method, an antibody is selected at random from the subset S_1 of the antibody population which contains those antibodies in which every job in the antibody has not yet been scheduled in the partially completed schedule. The new antibody is concatenated to the end of the partial schedule.

2. *Somatic Recombination* — in this method, an antibody is selected from the subset S_2 of antibodies, where S_2 consists of antibodies that overlap with the current partially completed schedule. An overlap is said to occur if the first n jobs in the antibody are equal to the last n jobs in the partially complete schedule, where $n \leq l$, and the remaining $(l-n)$ jobs in the antibody *do not* occur in the partial schedule. The partial schedule is thus extended by $(l-n)$ jobs.

- Until a schedule is complete *or* cannot be further extended:
  - Select a recombination mechanism with probability p_r for simple recombination, p_{sr} for somatic recombination and p_a for single job addition
  - Calculate the set S of possible antibodies that can be added via the chosen mechanism
  - Choose an antibody at random from S
  - Extend the partial schedule using the chosen antibody

Figure 3.17: An algorithm for generating completed schedules from the antibody population evolved used $SCHED2-IS$

3. *Single Job Addition* — in order that a complete schedule can be built when the antibody population does not contain at least one instance of each of the j jobs, a single job can be selected from the subset S_3 of all jobs that do not occur in any of the antibodies. This is then added to the end of the partial schedule.

The procedure for forming a schedule using these mechanisms is shown in figure 3.17.

3.12 Generating Test Data

Experimental test-data was generated in a manner similar to that described in section 3.6.1, using the same benchmark problem $jb11$. Ten test-scenarios were generated by mutating the original arrival date for each job with probability $p_u = 0.2$ to another random date, in the range (0,300). A satisfactory schedule was generated for each scenario using the genetic algorithm described by [Fang et al., 1993]. From each schedule (which consists of the order and timing of processing of every operation on every machine) 5 antigens can be generated, where each antigen corresponds to the sequence

of jobs on one of the 5 machines. Therefore, 5 antigen universes, one for each machine, are generated, and each universe contains 10 antigens, one derived from each test-scenario. The universes are treated independently, and thus an independent set of antibodies discovered for each machine.

3.13 Experimental Results

Experiments were performed to identify good settings for three main parameters; the antibody sample size σ, the antigen sample size τ, and the length of the antibody l. These initial experiments also attempted to answer several questions in order to assess whether the evolved antibodies would be useful pattern recognisers in the context of a scheduling system, for example:

- How many antigens are matched by at least one antibody ?
- How many unique antibodies are found ?
- What is the average number of antigens matched by an antibody ?
- How many actual jobs are represented in the set of antibody patterns ?

All reported experiments were performed using a population of size 100, with the length of each antibody in the population equal to 5 jobs, i.e 1/3 the length of the antigen string. The length of each antigen was 15, as each machine required 15 operations to be scheduled. Every experiment was run for 250 generations and was repeated 10 times. The mutation rate in each case was $1/l = 0.2$. Details of the settings for σ and τ are described under the relevant headings. As a separate set of antibodies are evolved for each machine, results are reported for only one of the machines, machine 1, however, the reported trends were applicable to all machines.

In the following section, many of the experiments are analysed with reference to a quantity described as the *match-threshold*. The match-function M described in section 3.11.3 quantifies the extent of the match between an antibody and an antigen. However, a qualitative statement regarding whether or not a match has occurred can be made — a match is only said to occur if the match-score M is greater than or equal to some

threshold value M_T. (A similar phenomenon is observed in natural systems in which an antibody does not bind to an antigen until the strength of the bond between the two — the antibody *affinity* — reaches a certain value). In this case, the match-threshold is more specifically defined as the number of non wild-card places in which an antigen and antibody match.

3.13.1 How many antigens are matched by at least one antibody ?

The aim of these experiments is to investigate the amount of *overlap* occurring in the antibody population, and also to determine whether or not all antigens are at least partially recognised. Table 3.4 shows the average number of antigens (from a universe containing 10 antigens) that were *not* matched by any antibody, for match-thresholds M_T ranging from 2 to 5. Experiments are performed over a range of values for σ and τ, the size of the antibody and antigen samples respectively.

For certain combinations of values of (m, σ, τ), it can be seen that *no* antibodies match some of the antigens. This is particularly noticeable as the size of τ increases. This is explained with reference to the fitness function used; for high values of τ, antibodies that successfully match more than one antigen are rewarded most highly. However, in many antigen-universes, it may be impossible to detect common patterns between certain subsets of antigen, and hence the completely generalist antibody may not exist. Examining the actual antigen universe for machine 1 indicates that for some subsets of antigen, common schemas do exist. Similarly, low values of σ also encourage generalist antibodies to evolve, so we may expect poor performance if the value of σ is too low for certain antigen universes.

3.13.2 How many unique antibodies are evolved ?

Recalling that the aim of $SCHED2-IS$ is to acquire a collection of antibodies, each of which represents some commonly occurring pattern in the antigen-universe, then the more unique patterns we are able to detect, the more useful the antibodies will be as building blocks for constructing new schedules. Therefore, the final population of antibodies is examined to determine the exact number of unique antibodies that match

Match Threshold	τ=1			τ=4			τ=8		
	σ			σ			σ		
	5	10	30	5	10	30	5	10	30
2	0.9	0	0.0	2.2	0.9	0.0	3.5	2.5	0.9
3	5.3	2.6	1.6	5.4	3.2	2.0	5.5	4.7	4.1
4	8.7	7.1	5.2	7.8	7.3	6.3	8.6	8.1	8.2
5	9.7	9.5	8.8	9.5	9.5	8.7	9.7	9.6	9.5

Table 3.4: Average number of antigens (out of a possible 10) not matched by *any* antibody

Antibody Sample Size, σ	Antigen Sample Size, τ			
	2	4	6	8
5	23.8	23.7	20.0	17.7
10	38.6	28.0	24.5	20.5
30	58.4	44.4	24.9	39.7

Table 3.5: The Number of Unique Antibodies in the Final Population for Match Thresholds ≥ 2

an antigen with a match-score $\geq M_T$, the match-threshold.

Table 3.5 shows the results obtained for $M_T \geq 2$. It is clear that the number of unique antibodies decreases as τ is increased, and increases as σ increases. This is unsurprising — the same arguments outlined in section 3.13.1 apply. Note however it is not desirable to produce an entire population of unique antibodies, and that having multiple copies of a matching antibody may ultimately be useful in the recombination phase when antibodies are selected and combined into schedules. The number of copies of an antibody in the population is somewhat analogous to a *concentration* in biological terms, and is an indication of the probability of picking the antibody when trying to reconstruct a schedule.

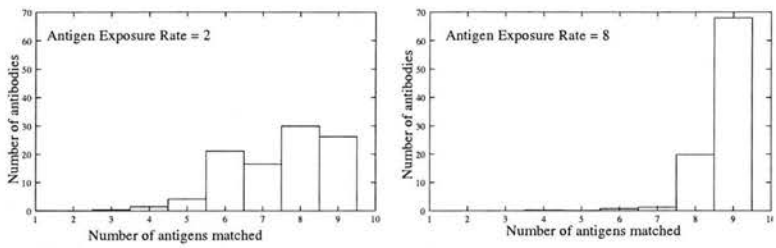

Figure 3.18: Match Threshold 2 : Number of antibodies matching > 1 antigen for antibody sample size = 30. The graphs compare the overlap when $\tau = 2$ and $\tau = 8$

3.13.3 Measuring Overlap

The two previous sections have shown that it is possible to evolve a set of unique antibodies, and also that those antibodies tend to match at least one antigen. In order to measure how much *overlap* is occurring between those antibodies however, the number of antigens matched by each antibody in the population is recorded. Figure 3.18 shows the number of antibodies that match n antigens, where n takes values between 1 and 10. The diagram contrasts the results obtained for a match-threshold of 2 and fixed antibody sample size $\sigma = 30$ for various values of τ. Clearly, more antigens are matched at high values of τ, as expected.

It is interesting to observe how the number of antibodies matching more than 1 antigen increases as the GA runs. The graphs in figure 3.19 illustrate the point. Experiments are performed in which the match-threshold m is varied from 2 to 5, for populations of antibodies in which the antibody length is varied from m to 5 also. When the match-threshold is equal to the antibody length, there is a rapid increase in the number of matching antibodies in the $m = 2$ case after which the number remains relatively constant. For $m = 3$, the number rises steadily. For values of $m < l$, in each case, there is an immediate increase at the the start of each run to a level which is maintained throughout the remainder of the experiment. The initial rise is more pronounced when m is significantly less than l.

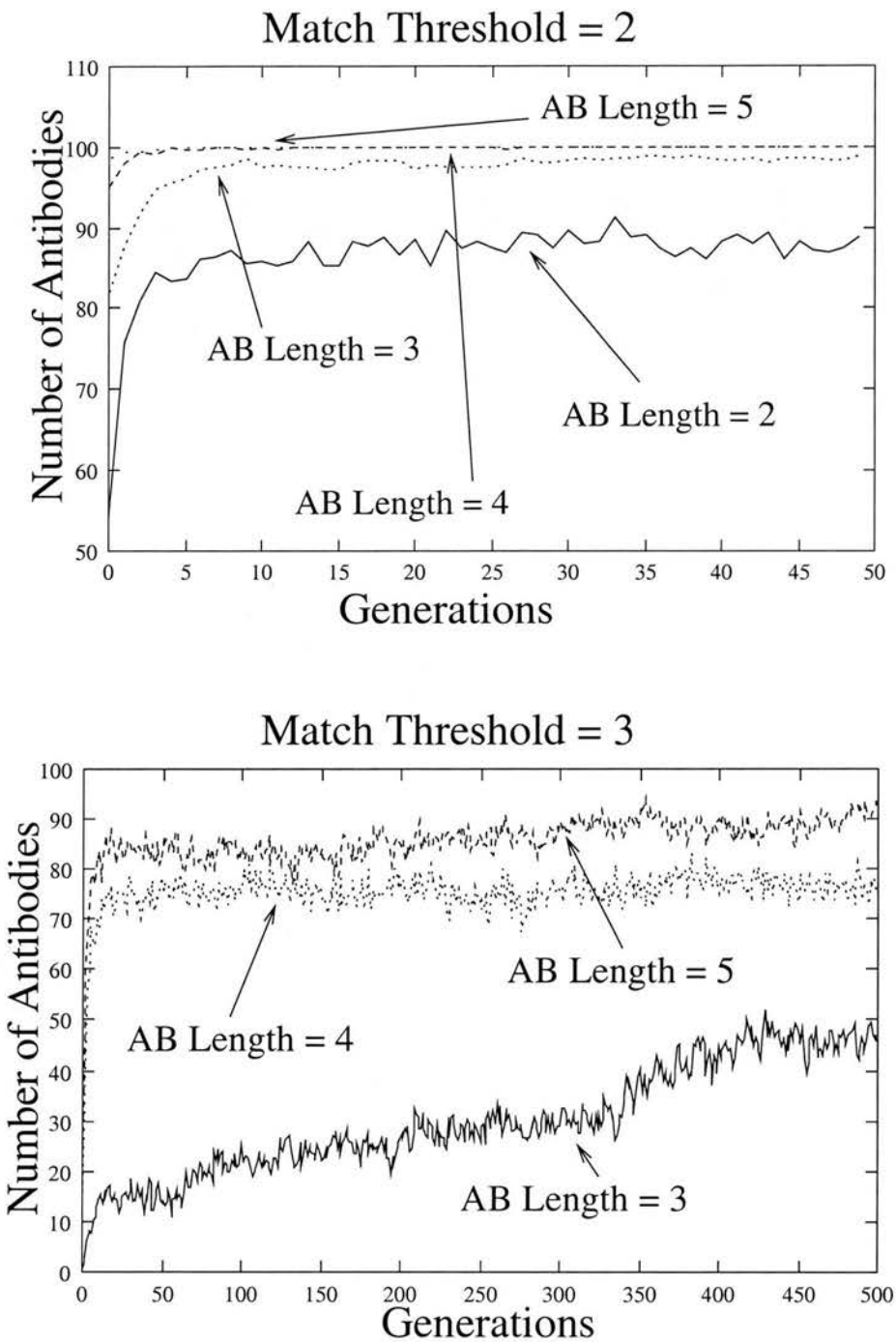

Figure 3.19: Number of antibodies matching more than 1 antigen for experiments in which the threshold for matching was set to 2 and 3

	Antibody Sample Size		
	5	10	30
Average Number of Jobs Represented	5.6	7.5	9.1

Table 3.6: Average Number of Jobs Represented in Final Population for Given Antibody Sample Size σ

3.13.4 Identifying the number of jobs appearing in the antibodies

The greater the number of *jobs* that are represented by the antibodies, the more useful the patterns will be in constructing new schedules; clearly the antibody building blocks will cover more parts of the schedule if this is the case. The exact number of jobs expected to occur in the final antibodies will depend of course on the nature of the antigen-universe; for some universes, there may only be a few jobs which belong to common pattern sequences. Recall that for machine 1, there are 15 possible jobs. Table 3.6 compares the number of unique jobs found in matching antibodies for 3 different values of σ, for match-thresholds ≥ 2. For small values of σ, very generalist antibodies evolve, representing only a few jobs. However, as σ is increased, the number of jobs represented increases as small clusters of more specialist patterns begin to evolve.

3.13.5 Reconstructing Schedules from the Antibody Population

Schedules are reconstructed from the antibodies evolved using $SCHED2-IS$ using the algorithm given in figure 3.17. The experiments just described suggest that the evolved antibodies appear to have good properties in terms of antigen coverage, overlap and redundancy — however, it is necessary to address the question of whether this set contains sufficient information to reconstruct good schedules, and if so, how effective is the proposed recombination algorithm.

It can be noted that for any antigen of length j, *all* antibodies of some predefined length l containing at most w wild-cards can be generated using an exhaustive procedure, without having to resort to using any kind of immune metaphor or genetic algorithm. For the problem described in which each universe consists of 10 antigens each containing 15 jobs, this is actually a tractable calculation. Performing the calcu-

lation to find all antibodies of length 5, that contain at most 3 wild-cards, results in 423 unique antibodies being found. This can be compared to the results shown in table 3.5, which shows that the *maximum* number of unique antibodies found in any experiment using $SCHED2-IS$ is 58 ($\tau = 2, \sigma = 30$). The evolved antibody set is thus clearly much smaller than the size of the set generated via exhaustive search. Therefore, experiments were conducted in order to determine the quality of schedule produced from the antibody set evolved using $SCHED2-IS$, and then repeated using the complete antibody set, so that a comparison could be made.

As with the experiments described to evaluate $SCHED1-IS$, two series of experiments were performed. The first tested whether the schedule present in the original antigen universe, $U(0.2)$, could be accurately regenerated. The second set of experiments investigated whether schedules in two new universes, $U(0.1)$ and $U(0.3)$ could be generated from the evolved antibodies.

In all experiments, the recombination algorithm was applied using the set of 58 antibodies generated when $\tau = 2, \sigma = 30$ which contain a maximum of 3 wild-cards, and then using the 423 antibodies generated by exhaustive search. In each case, 500 schedules were generated in response to each of the 10 scenarios in the original antigen universe. Trial investigations of suitable settings for the parameters of the recombination algorithm suggested that $p_r = 0.5$, $p_{sr} = 0.4$, $p_a = 0.1$ produced satisfactory results, although an exhaustive search of the parameter space was not performed. All experiments investigated the quality of reconstructed schedules starting from a partial schedule of length l_p, where $l_p \geq 0$.

3.13.5.1 Results in familiar antigen universes

The recombination algorithm performed very poorly when $l_p < 6$, in that *no* schedules were generated that exactly matched the original ones, using either set of antibodies. This is not fatal; as originally stated, it is expected that the utility of the system lies in rescheduling following an unforeseen circumstance, and *not* in constructing schedules from scratch, therefore, in a real situation, l_p is likely to be greater than 0. Table 3.7 shows the percentage of the 10 original schedules ($U(0.2)$) that were exactly reconstructed, and gives results for three values of l_p, the length of the partial schedule that

	Length of Partial Schedule			
	6	7	8	9
$SCHED2-IS$ antibodies	10%	30%	70%	80%
Exhaustively generated antibodies	0%	60%	60%	60%

Table 3.7: Accuracy of schedule reconstruction: The table shows the percentage of the 10 schedules in the original antigen universe ($U(0.2)$) that were exactly reconstructed using the recombination algorithm

must be completed. Results are shown using the evolved antibody set (58 antibodies) and the exhaustively generated set (423 antibodies). For $l_p = 8$ and $l_p = 9$, the best results are obtained when using the 58 $SCHED2-IS$ antibodies, though note that neither antibody input set is able to achieve 100% accuracy of reconstruction. However, when $l_p = 7$, (and hence there are $8! = 40,320$ possible combinations of the remaining jobs to be scheduled), using the larger number of antibodies generated by exhaustive search results in higher accuracy of reconstruction.

The *reliability* of schedule reconstruction can also be examined, i.e. the average number of original schedules that are generated each time the algorithm is run. (Recall that each pass of the algorithm produces 500 schedules per antigen). These results are shown in table 3.8. The reliability is rather poor, for both methods of generating the antibody set, and there is little difference in performance between sets. Further examination of the reliability with which individual antigens are reconstructed reveals significant variations, with some appearing to be easy to reconstruct (i.e. at least 20% of the 500 schedules are accurate), and others much more difficult (less than 1% of the 500 schedules accurate).

3.13.5.2 Performance in Unseen Universes

As with $SCHED1-IS$, two further antigen universes were generated as described in section 3.6.1 using $p_u = 0.1$ and $p_u = 0.3$. This produces universes identified as U(0.1) and U(0.3) respectively. Five antigens were generated in each case, and satisfactory schedules were again found using a GA as before. Schedules were again generated

	Length of Partial Schedule			
	6	7	8	9
$SCHED2-IS$ antibodies	0.32	1.75	5.53	30.79
Exhaustively generated antibodies	0	4.83	9.92	26.19

Table 3.8: Reliability of schedule reconstruction: The table shows the average number of times a schedule was accurately reconstructed during one pass of the recombination algorithm (which generates 500 schedules per antigen).

	Antibody Generation Method	
	$SCHED2-IS$	Exhaustive
Universe-0.1	17.2%	9.45%
Universe-0.3	32.76%	10.17%

Table 3.9: Antibody Recognition Rates in Unseen Universes: The table shows the percentage of the antibodies in each set (generated via $SCHED2-IS$ or exhaustively) that are able to bind to at least one antigen in each unseen universe

using the recombination algorithm applied to two sets of antibodies; those produced by $SCHED2-IS$ when applied to universe $U(0.2)$, and those found by exhaustive search of $U(0.2)$ that contained at most 3 wild cards, for values of l_p such that $0 \leq l_p \leq 10$. The results were disappointing — no accurate schedules were generated in any case for either universe. However, the fault appears to lie with the recombination algorithm itself — if the abilities of the two antibodies sets to actually match the new antigens are examined, then it is clear that the antibodies are capable of recognising the entirely new antigens in these universes. Table 3.9 shows the percentage of each antibody set that matches *at least one* of the antigens in the new universes. Clearly, the antibodies do encapsulate some useful information, i.e. they can bind to previously unseen antigens. The $SCHED2-IS$ antibody set is superior to that generated by exhaustive search in this respect. However, the recombination algorithm does not appear capable in its current form of combining these antibody segments in a useful manner into schedules that resemble those forming the original antigen universe.

3.14 Summary of Utility of $SCHED2-IS$

The previous sections have described the implementation of a new scheduling system based on a subset of biological immune system principles, extending a model first proposed by [Forrest et al., 1993]. Analysis of the effectiveness of the new model was split into two parts; a study of the fitness function used to evolve patterns or building blocks that could be used to reconstruct schedules, followed by an investigation of how these blocks could be recombined into schedules using mechanisms inspired by biological immunology.

The experiments detailed in section 3.13 show that in as far as producing building blocks that encapsulate useful information about the antigen universe in which they were evolved, the new model is successful. Extending the fitness function proposed by Forrest *et. al* to incorporate matching of multiple antigens (via the τ parameter) resulted in an algorithm that was capable of being tuned in order to produce antibodies that occur in schedules found in both familiar and unfamiliar antigen universes. In particular, the results given in section 3.13.5.2 which examines whether evolved antibodies can bind to antigens in new universes imply that the evolved antibodies are to some extent capturing some of the essential properties of the historical schedules. More importantly, this appears to validate the original assertions made in section 3.9 that the historical schedules *do* actually contain information that can be used to construct *new* schedules in unforeseen circumstances.

The results also demonstrate that the combination of a genetic algorithm/immune metaphor in the manner described is a suitable mechanism for performing pattern recognition in target datasets that consist of non-binary strings, and therefore that the generic model may be useful in other domains. In order to achieve these results, a new crossover operator was introduced in combination with a method which selected from a set of three possible crossover operators, according to the relationship between the chosen parent antibodies. This approach could be adopted in any domain in which antigens can be represented by categorical or integer alleles. Thus, as a simple pattern-recognition system, the model has many potential applications.

However, the second part of the analysis in which an attempt was made to mimic some of the mechanisms by which the biological immune system constructs antibodies

was less successful. Section 3.13.5.1 showed that satisfactory performance could be achieved in familiar universes, but that overall, performance was unreliable, especially when the partial schedule to be completed was short. It should be noted however that the reported work only examines one measure of quality of the reconstructed schedules, namely whether the ordering of operations on machines could be exactly reconstructed. It is possible that if other measures of quality were examined, for example, the difference in maximum job tardiness in a reconstructed schedule compared to the original schedule, then a rosier picture would emerge (or vice versa!). However, the fundamental difficulty with the proposed approach seems to lie in modelling an appropriate combination of recombination mechanisms, and using them with appropriate frequencies, relative to each other.

Reference to the immunological literature offers few clues in this respect — for example, [Roitt et al., 1988] describe how even though mammals in general may use at least five different mechanisms in order to produce diverse antibodies, different species rely on different subsets of these mechanisms. Thus, sharks rely solely on having very large numbers of genes which are randomly recombined, whereas chickens have very small numbers of antibody building blocks and rely heavily on gene-conversion to produce antibodies. The computational immune system literature has not addressed this aspect of immunology at all and is clearly an area in which much further research could be performed.

With regards to this study, further work could be performed in order to explore the space of parameters over which the model functions more reliably and hence optimise the choice of parameters, however there will always be an underlying trade-off between the number of antibodies required to adequately represent possible antigen universes, and the effort required to recombine them into accurate schedules. Furthermore, as just noted, the biological literature suggests that the 'right' set of parameters might vary for each problem encountered, which is clearly a major drawback for the system. If sensible heuristics to cut down the effort required to search the potential space of schedules that can be produced via these recombination methods cannot be found, then the object of producing a reduced set of schedule building blocks is somewhat defeated.

3.15 Conclusion

In conclusion, some useful *general* properties regarding the use of immune system analogies for tackling pattern recognition problems have emerged, from consideration of both $SCHED1-IS$ and $SCHED2-IS$. The work described has shown that both immune system models are appropriate for use with non-binary data, and that combining the immune metaphor with a genetic algorithm provides a workable mechanism for discovering patterns in diverse datasets. However, as a practical proposition for solving real-world scheduling problems, both models exhibit weaknesses. In each model, the major weakness lies in the methods proposed for combining the evolved antibody building blocks into complete schedules. Despite having evolved a reduced set of useful building blocks (whether stored in libraries as in $SCHED1-IS$ or as a population in $SCHED2-IS$) the combinatorics of joining those blocks into suitable schedules quickly becomes intractable, even when the mechanisms used by the biological immune system itself are modelled. Furthermore, experiments presented here only consider relatively short schedules; in real life, schedules are likely to be much longer and more varied, and hence the difficulties will become even more exaggerated.

The biological immune system is able to circumvent these difficulties due to its implicit massive parallelism; an individual has sufficient genetic material to express in the order of $> 10^{10}$ distinct antibodies, and at any one time, a subset of between 10^7 and 10^8 antibodies are expressed. These antibodies circulate throughout the body and are continually replaced if they do not encounter antigen to which they can bind. Therefore the problem of producing the 'correct' antibody to eliminate an invading antigen is tackled by sheer size and scale. Clearly, this effect cannot be mimicked in a computational system, even using todays massively parallel computers. This begs the question therefore as to whether the fundamental approach of evolving useful building blocks which can be cleverly combined into larger pieces is flawed; if the biological immune system does indeed offer metaphors which are useful for computational information processing, then perhaps these models have concentrated on the 'wrong' subset of the metaphors.

Consider again the list of key features of the immune system important to the field of information processing outlined by [Dasgupta, 1998] and given in section 1.2, chap-

ter 1, of this thesis. The foremost aspects of the biological immune system modelled by $SCHED1-IS$ and $SCHED2-IS$ are *memory* and *learning*. Both artificial systems store memories of past experiences, which they have learned, and can generate responses to new and novel patterns. In order to do this, they also draw on the *diversity* mechanisms apparent in the biological immune system, and make use of various *recognition* mechanisms. However, several key features of the biological immune system are *not* modelled in the two systems described.

Two of the features of the biological immune system which distinguish it from other biological metaphors which have been borrowed by computer scientists are its *self-regulatory* nature, and its inherent ability to operate in a *dynamic* environment in a manner which is itself dynamic and *adaptable*. The systems proposed here are not in any sense self-regulating; they are controlled via a genetic algorithm with an explicit fitness function. Furthermore, although the *environment* in which they are designed to operate, i.e. a scheduling scenario, is dynamic, the *systems* themselves are not dynamic; in each case, a static set of antibody building blocks is evolved, and the processes governing recombination of these building blocks do not incorporate any feedback mechanisms which could direct the recombination to *currently* useful parts of the space of possible antibodies as in the biological immune system. Another property not explicitly modelled by either $SCHED1-IS$ and $SCHED2-IS$ is the distributed nature of the biological immune system, although in theory both of these models could be implemented in a distributed manner. These factors point to the fact that perhaps other models of the biological immune system could be derived which incorporate these important principles. Ultimately, modelling these features may prove that the immune metaphor can provide something which other biologically based algorithms cannot, whereas currently, the argument in favour of the AIS is far from emphatic.

Although the introduction to this chapter pointed out many theoretical similarities between the scheduling environment and that in which the biological immune system operates which originally led to the presumption that it would be a suitable area in which to apply the immune system metaphor, there is also one major difference, namely the size of the antigen space in which the systems operate. It has been suggested that theoretically there are between 10^{12} to 10^{16} possible antigens that could

invade the biological immune system. This is clearly much larger than the number of possible scenarios that could *realistically* arise in a typical scheduling environment, even when considering very large schedules and the remotest of situations arising. Thus, given that the important features of the immune system presumably arose in order to allow it to operate in such large environments, perhaps it would be more fruitful to direct research into computational implementations of immunological metaphors towards application areas in which the environment was larger and more diverse. Those areas most suited would also have a dynamic nature, as with scheduling, in order to maximise the utility of the metaphor.

Thus, in the remainder of this thesis, we take a step back from the real-world, and return to more basic principles, in order to try and derive an immune-based model that incorporates the fundamental features of the biological immune system, and that can be applied in environments whose features more closely resemble those of the biological world. Two novel models are proposed in the following chapters — the models are tested with artificially generated binary datasets whose properties can be exactly controlled, as in the majority of the seminal work on library and population based artificial immune systems. This allows a more detailed analysis of the performance and potential of the new models. However, both models theoretically could be extended to allow more realistic problems to be tackled.

Chapter 4

Applying an Immune System Analogy to Data-Clustering Problems

4.1 Introduction

The previous chapter on applying immune models to scheduling problems uncovered some deficiencies of the immuno-genetic approach to modelling aspects of the biological immune system. In particular, section 3.15 of chapter 3 referred to some of the important properties of the biological immune system not modelled by coupling a genetic algorithm with a library-based or population-based approach to pattern recognition, namely distributed detection, self-regulation, and dynamic protection. Furthermore, although the models outlined for scheduling incorporated some elements of *probabilistic detection*, another of the key features of the biological immune system, this was not an implicit feature of either model. The conclusion of the preceding chapter also stated that in order to maximise the utility of the metaphor, then the applications to which the metaphor was applied should be chosen carefully; ideally an application that encompasses a domain that is *dynamic* and potentially contains very large numbers of data or antigens would prove most suitable.

In this chapter, it is proposed that an ideal testbed, which contains all the desired properties, is that of data-clustering in large, real databases. The reasons underlying this are explained in the following sections, which identify the relationship between a

data-clustering system and an artificial immune system. A further system, the *Sparse Distributed Memory*, SDM, which has already been shown to be analogous to the AIS is then introduced in more depth. It is then shown how aspects of the SDM and AIS can be combined in order to derive a model theoretically capable of performing data-clustering. Finally, the remainder of the chapter lays the foundations for generating an artificial data testbed which is used for testing two novel immune-based models, described in chapters 5 and 6.

4.1.1 Data-Clustering with an Artificial Immune System

Consider a database to represent an antigen universe: a single item of data in the database represents an antigen that must be recognised by the host system. An antibody produced by the artificial immune system recognises a set of antigens in the antigen universe — in a data clustering context, those items recognised by a single antibody can be considered to form a cluster, and the antibody itself represents a concise description of that cluster. In biological terms, the affinity or size of the ball of stimulation of the antibody would determine the size of the cluster. Assuming all antigens in the universe can be recognised by the antibody set, then the number of antibodies present determines the generality/specificity of the clusters; a small number of antibodies will result in few clusters, and therefore each cluster represents a very general description of the data. As the number of antibodies is increased, the specificity of the cluster and hence the concept it represents also increases. New data arriving in the data-base is continuously presented to the system, which triggers the antibody set to adapt to the new dataset, either by adapting existing antibodies, or creating new ones. This is exactly analogous to the primary response in the biological immune system.

4.1.2 Data-clustering with a Sparse Distributed Memory

As previously stated in chapter 2, [Smith et al., 1996] showed that immunological memory belongs to the same class of associative memories as Kanerva's Sparse Distributed Memory (SDM). In this work, they argued that the B-Cells and T-Cells of the immune system perform the same function as the hard locations in an SDM, and

showed that the hard locations in an SDM provide a sparse coverage of possible addresses in the same manner as the B-Cells and T-Cells of the immune system provide a sparse coverage of all possible antigens. Therefore, it ought to be possible to apply this analogy to data-clustering problems, in the same manner as described in the previous paragraph.

Thus, in this case we can assume that a hard-location in the SDM recognises a subset of data-items in a database based on a comparison between the physical address of the hard-location and the data-item itself. If the measured distance between a hard-location and a data-item is within the recognition radius of the hard-location, then recognition of the data occurs, and the data is stored at that location, updating the counters of the location in the process. All data recognised by a particular hard location can be considered to lie within the same cluster — the physical address of the hard location represents a description of the cluster. A benefit of this approach is that the counters associated with each hard location also contain useful information — for example in binary systems, the absolute value of each counter can be interpreted as a probability of the particular bit within any data-item in the cluster being set to 1/0 and therefore supplies further information which can be used to describe the cluster.

The accuracy with which an SDM has clustered a set of data can be measured by attempting to *read* back from the SDM all data stored within it; the recalled data can then be compared to that actually stored in the database and the average accuracy of recall will indicate the accuracy with which the data has been clustered. (Note though, that as this is a form of *unsupervised* clustering, no labels are associated with each data-item and hence we are not concerned with determinining whether data has been correctly assigned to pre-defined classes).

The SDM models one of the features observed in the biological IS and particularly relevant for data-clustering; it is high likely in any real-world database that clusters will overlap. The SDM is a distributed system in which data can be stored at more than one location, and in which more than one location may be involved in recall of that data. (The locations and recognition radii define the extent to which clusters will overlap). Thus there is an interaction between hard locations, just as in the biological immune system the specificity of a biological anti-serum is a function of a number of

interacting antibodies, and not simply a result of a single antibody reacting exclusively with the inducing antigen [Roitt et al., 1988]. This feature of biological immune systems does not appear to have been explored in any of the work reviewed in chapter 2 which shows that there has been a tendency to concentrate on models which produce an artificial anti-serum containing a set of co-operating but non-interacting antibodies, [Smith et al., 1993, Potter and De Jong, 2000].

4.1.3 Properties of the SDM/IS Models Relevant to Data Clustering

The previous two sections have identified how both the IS and SDM analogies could theoretically be applied in a data-clustering context. We now attempt to draw together the salient features of both models that suggest this, before identifying the drawbacks in both models that need to be overcome.

- Both the IS and SDM provide sparse coverage of very large input spaces — this is desirable in a data-clustering environment as we wish to identify *small* numbers of clusters in very *large* datasets.
- The IS and SDM operate using an imprecise recognition mechanism — real data is likely to be incomplete and/or incorrect, in addition to containing superfluous noise, and therefore a system which can perform recognition under these conditions is necessary.
- The IS and SDM are distributed systems, and therefore robust to loss of pattern detectors. As very large datasets are also likely to be distributed, a model based on an SDM/IS architecture appears suitable.

Other properties also emerge — for example, both the SDM and IS models implicitly contain a mechanism for detecting *change* within an environment, and therefore in the context of data-clustering, for detecting data which does not belong to an already identified cluster. For example, in an IS-based clustering model, if a data-item is not recognised by one of the antibodies defining a cluster, then the data can be flagged as unusual, perhaps signifying that it is non-representative of the general patterns and therefore triggering some warning. Imagine for example attempting to cluster data

collected by a credit-card company relating to card usage. The company is interested in clustering the data to identify patterns in card usage, but would also like to detect fraudulent card-usage. If a newly presented data-item does not belong to an already established cluster, it could identify an attempt at fraudulent usage of the card, which further human examination could identify,

In the *danger model* of the IS, expounded by Matzinger in [Matzinger, 1994b], a dynamically adapting system such as the IS would adapt over time to contain antibodies that recognised both 'normal' and 'abnormal' clusters (as opposed to the negative selection model of [Percus et al., 1993], modelled artificially for example by [Forrest et al., 1994, Dasgupta and Forrest, 1995, Hofmeyr and Forrest, 2000], in which the IS only contains antibodies that recognise abnormal data). In an artificial model adopting the danger approach, and therefore recognising and clustering *all* data, the rate and extent to which the antibody set changes gives some indication of the extent to which the system has to adapt to cluster new data, and therefore could also be used to identify the occurrence of new and unusual trends in the data. The 'abnormal' data is clustered, and hence it is straightforward from then on to recognise when new data falls into one of these clusters. The SDM class of associative memories appears to fall into the danger-model camp of immune system models, in that it is capable of recognising and storing *all* data present in a given environment, and not just data that falls into either the 'self' *or* the 'non-self' categories.

4.1.4 Inadequacies of the SDM

Although the previous sections have shown that the SDM appears to be an ideal analogy on which to model a data-clustering system, and that [Smith et al., 1996] have discussed in detail the close relationship between the immune system there are also some fundamental differences between the two systems. Moreover, there are several flaws in the fundamental postulates defining Kanerva's original SDM that must be modified to produce a practical system for use in a data-clustering environment.

Three of the original postulates underlying the original SDM are as follows:

1. The number of hard storage locations and their addresses is known from the start, and only the contents of the locations are modifiable.

2. The recognition radii are fixed from the start, and it is assumed that each location has an identical recognition radius

3. The hard storage locations are distributed randomly in the $0, 1^n$ address space

These postulates result in a memory which is in inflexible and inadequate for storing data which is not random and which is subject to change. Clearly this applies to real-world databases which are obviously non-random. In particular, postulate (3) which requires the storage locations to be randomly located will not only result in in a highly inefficient memory but moreover, cannot result in an accurately clustered database.

Furthermore, the size of real data-sets will vary over time as data is collected and removed, and also, the contents of the data will change with time. Therefore any memory which requires a fixed number of static hard locations will also be inadequate to perform the clustering task. This is clearly a fundamental difference between the biological immune system and an SDM — in the immune system, antibodies are continually produced by lymph nodes, and have a finite lifetime. Thus the number of antibodies circulating throughout the immune system varies with time, and the type of those antibodies also varies, depending on the current state of the antigen environment.

Therefore, we propose that a new model for performing data-clustering can be devised by integrating some of the dynamic and adaptive features of the immune system with the simple storage and associative recall ideas of the SDM. The basic premise underlying the model is that an immune system based on a dynamic SDM can be constructed, in which the number of hard locations is variable, and the location and recognition radii of those locations adapts to suit the environment to which the SDM is exposed at any particular time. Thus, to rephrase this using immunological terminolgy, the hard-locations in an SDM can be represented by antibodies in an immune system, and the data which it is hoped to cluster an antigens. The recognition radius in the SDM is replaced by a threshold mechanism in the immune model, below which binding between the antibody and the antigen does not occur. This is consistent with the mapping between the SDM and immune system originally proposed by [Smith et al., 1999] and described in chapter 2, in table 2.1. For consistency, the immunological terminology is used throughout the remainder of this thesis.

The remainder of this thesis describes two different methods by which an immunological model can be implemented. In chapter 5.2 we describe a model dubbed *COSDM* which uses a co-evolutionary algorithm as the mechanism for driving the discovery of antibodies and their associated binding thresholds. Chapter 6 adopts an approach named SOSDM based on simple self-organising algorithms such as that of [Kohonen, 1982b]. Both approaches are tested on artificially generated binary datasets, representing both static and non-static data-clustering problems in order to analyse their performance in detail and establish whether or not the proposed methods provide a starting point for tackling more complex real-world tasks. The generation of these datasets is now discussed.

4.2 Problem Description

Both the COSDM and SOSDM algorithms are first tested in a simple test environment which is identical to that used by [Potter and De Jong, 2000], and in [Forrest et al., 1993] in a related study, and therefore allows for straightforward comparison of the models. The problem is a binary string covering exercise — in its original most general form, it consists of finding a set of K binary strings (the *match-set*) that match as strongly as possible another set of N binary strings (the *target set*), where $N \gg K$. Thus, each of K strings in the match-set must contain a pattern(s) common to a subset of patterns in N in order to cover the set optimally, and therefore represents a generic description of some subset of N. The task therefore is to discover the best possible set of K strings — each of the K strings can be considered to represent a *cluster* in the original dataset, and if there are K clusters present, then K strings need to be discovered.

4.2.1 Stationary Data

Throughout the remainder of this thesis, three categories of antigen datasets are used for experimentation with the proposed models and for comparison with the algorithm described by Potter *et.al* , and from now on referred to as CE-POTTER (*C*o-*E*volution-POTTER). This method of generating data-sets is described in detail in

[Potter and De Jong, 2000]. The categories are generated in the following manner:

The first category of data is referred to as half-length data, and datasets generated in this category contain N antigens, each of length L. The data is generated in equal proportion from two half-length schemata. Schema-1 has the first $L/2$ bits fixed to 1, and the remaining $L/2$ bits contain wild-cards. In schema-2, the first $L/2$ bits contain wild-cards and the remaining $L/2$ bits are fixed to 1s. Therefore, if $L = 32$, the two schema are:

```
11111111111111################
################1111111111111111
```

Thus a dataset generated in this fashion will contain 2 clusters.

The second category of data is referred to as quarter-length schema and datasets are generated in equal proportion from 4 schemata, and therefore contain 4 clusters. The length of the defined section in each schema in this case is $L/4$. Using the same example as above in which $L = 32$ the schema will be as follows:

```
11111111##########################
########11111111################
################11111111########
########################11111111
```

Finally, the third category of data, eighth-length schema, produces datasets generated in equal proportions from 8 schemata, each with defined length $L/8$. Thus for $L = 32$, the 8 schema which are used to generate 8 clusters are shown below:

```
1111############################
####1111########################
########1111####################
############1111################
################1111############
####################1111########
########################1111####
############################1111
```

The length of the defined section in each case is denoted by d.

4.2.2 Non-Stationary Data

Section 4.2.1 describes how N antigens can be generated from s schema in equal proportions. All experiments using static data are generated from one of the three categories of data described; half-length schema, quarter-length schema and eighth-length schema. Each of these three categories contains non-overlapping schema. A more general method of generating data is to also generate the *schema* randomly, by choosing a random start point along the string and then setting d contiguous bits to 1. All remaining $(L-d)$ bit positions contain wild-cards. Schemas are more likely to overlap in this case, but this is likely to be a more accurate reflection of real data sets. This method can easily be adapted to produce non-stationary data sets using the algorithm shown in figure 4.1. Generating datasets in this manner enables the proposed algorithms to be evaluated in the context of the following properties of the datasets:

1. The number of clusters present in the dataset
2. The length of the defined section of each cluster
3. The extent of overlap of the clusters
4. The rate of change of the dataset in terms of the number of clusters replaced at each update, and the rate at which the update occurs.

This will enable some conclusions to be drawn about the suitability of the suggested approaches for clustering real-world datasets, in which sensible estimates can be made of the likely rate of change of the data, and also of the characteristics of the dataset.

4.2.2.1 Relevance of Data Generation Approach to Real Datasets

In a real dataset, we would expect to observe clusters of data, in which items within each cluster share common features. However, it is extremely likely that at least some of these clusters will not be distinct but will overlap with each other. Therefore, when designing an artificial dataset in order to test the proposed models it is essential that the datasets should exhibit at least two characteristics if performance on them is to be indicative of real-world problems:

1. Generate s schemas at random, each containing d contiguous defined bits, and $(L-d)$ wildcards
2. Generate (N/s) antigens from each schema
3. Every U time-steps:
   (a) kill of g randomly chosen schemas and their associated antigens
   (b) generate g new schemas and (N/s) new antigens from each new schema
   (c) add the new antigens to the dataset

Figure 4.1: A generic algorithm for generating non-static datasets

1. Data should form clusters, in which items within the cluster contain at least one shared feature, but in general are non-identical
2. Some or all of these clusters may overlap, i.e the data should contain some features that are shared by more than one cluster.

Generating antigens from schema in the manner just described fulfills these two criteria; the defined section of each schema corresponds to a feature in the data, and generating the remaining portion of each antigen from wild-cards ensures that there is diversity within the cluster. Moreover, the generic method of generating antigen from random *schema* described in section 4.2.2 also ensures that clusters can overlap.

In fact, this has a direct parallel in the biological immune system in which anti-serum raised against a set of antigen exhibits two phenomena — that of *specific reactions* and of *cross-reactivity*. The specificity of a biological anti-serum is equal to the *sum* of the actions of every antibody in the serum. Consider figure 4.2, taken from [Roitt et al., 1988], which shows a population of three antibodies; some antibody-antigen reactions are highly specific — this is shown in figure 4.2(a) in which individual antibodies are directed against specific epitopes (X,Y,Z) on three different antigen

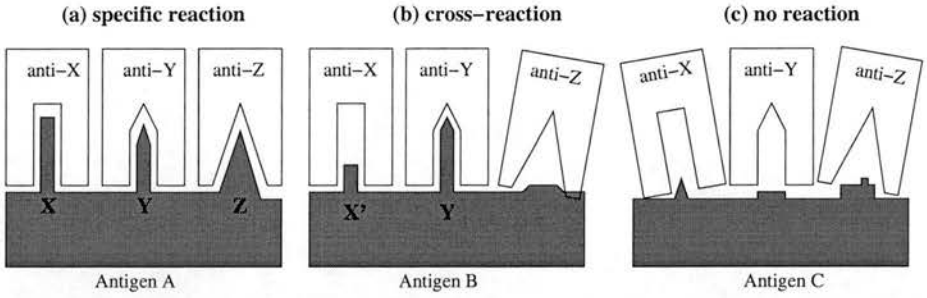

Figure 4.2: Anti-serum specificity results from a population of interacting antibodies. The figure illustrates three phenomena — specific reaction of antibody against individual antigen, cross-reactivity of antibody against more than one antigen, and no reaction

molecules. On the other hand, if an epitope is common to more than one antigen, for example epitope Y in figure 4.2(b), the antibody raised against X will *cross-react* with antigen Y. This figure also shows that the match between antigen epitope and antibody paratope does not have to be exact — antibody X is also capable of recognising epitope X'. Figure 4.2(c) also illustrates that this anti-serum has no reaction with antigen C, as no epitopes are able to be recognised. Relating this to a clustering problem, recognition of individual clusters is equivalent to the *specific* reaction of an antibody, whilst the phenomenon of *cross-reactivity* is modelled by recognition of common features in overlapping clusters.

The next chapter describes a co-evolutionary approach to evolving an SDM to perform data clustering on static and non-static datasets using data generated in the manner just described. Chapter 6 then describes a self-organising SDM and the results of experiments repeated on similar datasets.

Chapter 5

EA Based Model — COSDM

5.1 Combining Co-evolution with an SDM — COSDM

This chapter describes a new immune system model named COSDM — *co-evolutionary SDM* — which exploits the analogy between an SDM and the immune system to performs data-clustering. We propose to use a genetic algorithm within a co-evolutionary architecture in order to find a set of antibodies which accurately cluster a set of data. The immune system formed by these antibodies is analagous to an SDM made up of a set of hard locations. The size of each antibody's corresponding ball of stimulation (i.e. its recognition radius in SDM terminology), and the optimal number of antibodies required is evolved by the architecture.

The COSDM architecture is based on that described in [Potter and De Jong, 2000] which is a generalised architecture suitable for finding coadapted subcomponents or solutions to problems that can be decomposed into simpler subtasks between which they may be complex interdependencies. A model of this architecture is given in figure 5.1. The architecture models an ecosystem consisting of two or more species — as in nature the species are genetically isolated and therefore individuals within one species do not mate with others outside of their species. The species interact with one another however via a shared domain model, and as such have a cooperative relationship.

Figure 5.1 shows an example of a system which contains three species, each evolving within its own population via the application of a GA. The diagram shows how

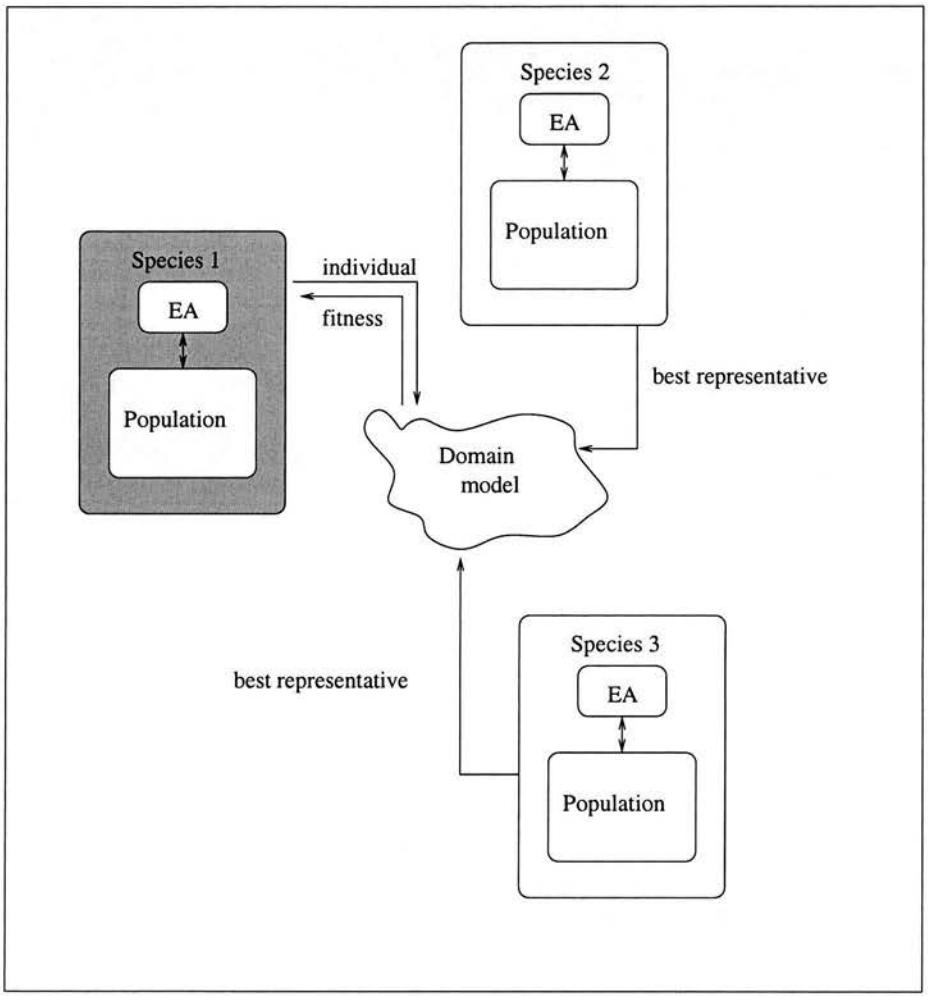

Figure 5.1: Coevolutionary architecture of the Potter model

the fitness evaluation proceeds from the perspective of one of the three species — to evaluate an individual from one species, collaborations are formed with representatives from each of the other species. Thus, in the case of the string covering problem described in chapter 4, individuals in each population represent match strings, and each species contributes one string to the domain model, i.e. the match-set. At any time, the match-set consists of the string under evaluation, plus the current best string from the other (N-1) populations. The match strength between a match-string $\vec{x}$ and an antigen string $\vec{y}$ is simply given by summing the number of bits in the same position with the same value, i,e:

$$S(\vec{x},\vec{y}) = \sum_{i=1}^{i=L} \begin{cases} 1 & if \quad x_i = y_i \\ 0 & otherwise \end{cases}$$

Then, to compute the match-strength of the set M, and therefore the fitness of the match-set to be assigned back to an individual, the match-strength is calculated between each of the N members of the set and each of the K antigens in the target set, and then the maximum computed strengths with respect to each antigen are averaged:

$$S(M) = \frac{1}{K} \sum_{i=1}^{K} max(S(\vec{m}_1,\vec{t}_i),...,S(\vec{m}_N,\vec{t}_i))$$

The architecture described by Potter lends itself well to implementing an immune system based on an SDM model that is able to successfully store and retrieve large amounts of data. Whilst the obvious subtasks in COSDM are to locate the best position and recognition radius of each antibody, there are also complex interactions between individual antibodies as counters are summed across the entire immune system in order to retrieve a piece of data. These interactions are handled well by the cooperative nature of the architecture.

A model of the *adapted* architecture, COSDM, is shown in figure 5.2. As in the original architecture, multiple populations are evolved. In the case of COSDM, each population in isolation controls the identity and recognition radius of potential antibodies. A complete immune system is formed by each population contributing one antibody to a *serum* which is then evaluated in order to determine how accurately it can store and recall data currently visible to the system. The serum consists of a member of the population currently under evaluation, and the best member of each of the

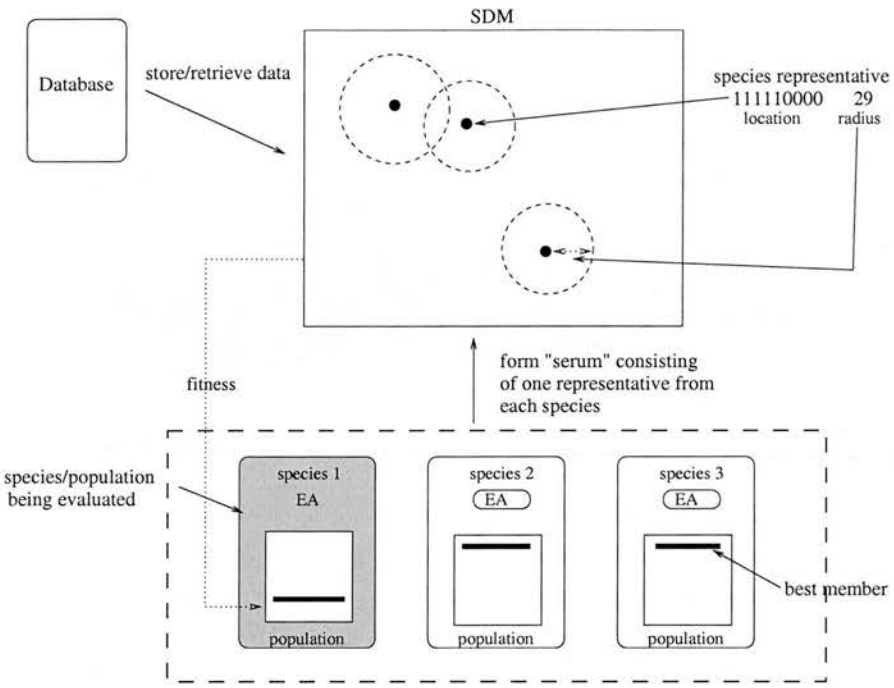

Figure 5.2: Coevolutionary architecture of the COSDM model

other population. Credit is then assigned back to the population under evaluation, and the usual mechanisms of selection and reproduction then take place within each population. In the implementation of the model described, populations are evaluated in a serial fashion, but there is no intrinsic barrier to performing a parallel evaluation.

Potter points out that there are four issues to be addressed in trying to produce an evolving computational model that provides reasonable opportunities for emergence of coadapted subcomponents of a larger problem. The properties of the proposed architecture that are desirable in the context of evolving an immune system for data recognition are now discussed. For a detailed explanation of other characteristics of the architecture, the reader is referred to the original paper, [Potter and De Jong, 2000].

Problem Decomposition When searching for the optimal immune system to store a large dataset, it is impossible to know *a priori* how many antibodies are required, particularly if the data is non-stationary. Therefore, the model should allow for addition and deletion of antibodies as the adaption takes place. This is simple to

achieve in the CE-POTTER architecture by adapting the mechanisms for adding and deleting subcomponents. The method by which this is implemented is described further in section 5.2.2.

Interdependent subcomponents As already identified above, the antibodies will exhibit a high degree of interdependency — COSDM is a distributed system in which antigen bind to *many* antibodies during both the storage and retrieval phases. Therefore, an antibody cannot be evolved in isolation. This kind of interdependent relationship is also observed in the biological immune system, where the specificity of an anti-serum is a function of a number of *interacting* antibodies and not a result of a single antibody reacting exclusively with the inducing antigen [Roitt et al., 1988]. The architecture described handles interdependent components as the evaluation phase evaluates one species in the context of all the other species by forming the immune system — it is impossible to evaluate a species in isolation.

Credit assignment When tackling problems that have have been broken down into subcomponents, the issue of distributing credit to each of the components always arises. The mechanism described by Potter which is used in order to determine the fitness of members of one species is to evaluate those individuals in conjunction with the best members of each of the other species, i.e. the individuals contributing to the serum from the other species remain fixed. The resulting fitness of the entire serum is then assigned to the individual being evaluated alone. This is particularly applicable to credit assignment for the immune system, where it would be very difficult to determine exactly how much a single antibody contributed to the overall fitness of the immune system.

Diversity Clearly, the repeated application of an EA to a population will eventually result in the convergence of that population. As fitness in a population is a result of collaboration and cooperation between all subcomponents, the architecture described should enable sufficient diversity to be maintained in each species until the *complete* problem has been solved. Furthermore, in a non-stationary system, there is a requirement to maintain sufficient diversity within the populations to

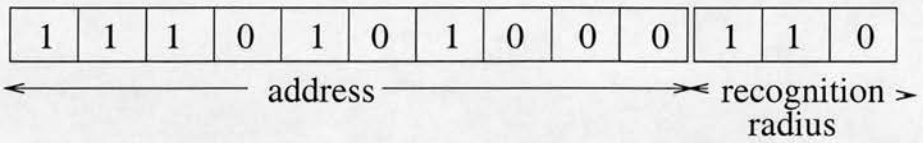

Figure 5.3: Structure of an antibody representing a hard location in an immune system. Each species in the COSDM consists of p such antibodies. Note that each antibody has an associated set of counters, one for each address bit, but that the counters are *not* evolved.

allow them to adapt to moving datasets. Whether the use of an EA will enable this to happen remains to be seen at this point.

5.2 Implementation of COSDM

As shown in figure 5.2, the COSDM architecture is assumed to contain n populations, with each species containing p potential antibodies. An antibody represents a complete description of a hard location in the immune system, that is it contains L bits representing the address of the hard locations, and a further R bits representing the recognition radius, ρ, of the location. Each antibody also has an associated set of counters (integers) — these are not evolved but are set by storing data in the immune system. An example antibody is shown in figure 5.3.

The address of a location defined by an antibody c is denoted by $V(c)$, and the counters by $C(c)$. R bits represent a Gray-coded description of the recognition radius. The actual algorithm defining the architecture and governing the evolution of each species is described in pseudo-code in fig 5.4. The key steps are described in more detail in the following sections.

5.2.1 Calculation of Fitness

Step 2 in figure 5.4 calculates the fitness of each individual antibody in each population, based on an evaluation of how that antibody i^* performs in the context of an immune system formed by itself and the *best* member of each of the other populations, referred

1. Randomly generate n populations, each containing p antibodies
2. Calculate the fitness of each member of each populations using the data currently visible to the system
3. Sort each population by fitness
4. Calculate the mean recall accuracy of the dataset using the immune system composed of the best member of each species $\bar{r}$
5. If $\bar{r}$ has not improved by at least τ over ϕ generations, add a new population
6. If the best member of any population does not recognise at least ε antigens, and the population has existed for at least λ generations, kill the population
7. Apply an EA to each population in turn in order to reproduce it
8. Go back to step 2

Figure 5.4: The COSDM algorithm

to from here on as SDM^*. (Note that at generation 0, when evaluating any single population, the best member of each of the other non-evaluated populations cannot yet be calculated, therefore a member of each of these populations is simply chosen at random to form SDM^*.)

The calculation takes place in two phases. In the first phase, a subset of the antigens in the dataset of size s, ($s \leq N$) is stored in the immune system represented by SDM^*. In the second phase, recall of the entire dataset is performed, using the counter values now contained in SDM^*.

5.2.1.1 Phase 1: Storage Phase

- Set the counters of each of the c_i $(1 \leq i \leq n)$ antibodies in SDM^* to 0s
- Present each antigen in a in subset s to the immune system:
  - Calculate the subset of antibodies n' for which the distance between the address of the antibody and the antigen, i.e. $H(c_i, a)$, is less than the recognition radius of the anibody, ρ_i, (equation 5.1). $H(c_i, a)$ is simply the Hamming Distance between the two strings $V(c_i)$ and a.

$$H(c_i, a) = \sum_{j=1}^{j=L} \begin{cases} 1 & if \quad V(c_i)_j \neq a_j \\ 0 & otherwise \end{cases} \tag{5.1}$$

  - For all antibodies in n', update the counter values at each bit in each antibody, according to equation 5.2:

$$\begin{aligned} &\forall j : (1 \leq j \leq L) : \\ &if\ H(c_i, a) \leq \rho_i : C(c_{ij}) \rightarrow C(c_{ij}) + \begin{cases} 1 & if \quad a_j = 1 \\ -1 & if \quad a_j = 0 \end{cases} \end{aligned} \tag{5.2}$$

5.2.1.2 Phase 2: Recall Phase

For each antigen a in the dataset:

- Calculate the subset of antibodies n' for which the distance $H(c_i, a)$ is less than or equal to ρ_i according to equation 5.1.
- Sum the counters of each member of n' at each of the j bit positions to give $\sigma_j(a)$ at each position:

These values are then used to calculate the *actual* recalled bit, a'_j for each of the L bits in the antigen:

$$a'_j = \begin{cases} 1 & if \quad \sigma_j > 0 \\ 0 & if \quad \sigma_j < 0 \\ (0,1) & randomly\, chosen\, otherwise \end{cases} \tag{5.3}$$

The recalled bit-string a' can then be compared to the actual antigen originally stored in the memory a, and the *match-score*, $\mathcal{M}$ calculated. This is simply the number of bit positions in which $a' = a$.

$$\mathcal{M}(a', a) = \sum_{j=1}^{j=L} \begin{cases} 1 & if \quad a'_j = a_j \\ 0 & otherwise \end{cases} \tag{5.4}$$

The mean recalled accuracy of the entire dataset $\bar{r}$ is then simply the average match-score obtained for each of the recalled antigens:

$$Mean\, recall\, accuracy\, \bar{r} = \frac{1}{N} \sum_{i=1}^{i=N} \mathcal{M}(a'_i, a_i) \tag{5.5}$$

Thus the fitness of SDM^*, and hence that of i^*, the individual under evaluation, is simply equivalent to $\bar{r}$.

In equation 5.3, randomly choosing the value of the recalled bit when $\sigma'_j = 0$ introduces a concept similar to the *somatic mutation* that is observed in the real immune system (see chapter 2). However, an alternative approach which would perhaps result in a more stable system would simply be to copy the bit from the address whenever $\sigma'_j = 0$, i.e. $a'_j = V(c_i)_j$.

5.2.2 Control of Number of Species

The number of antibodies in the final system is dynamic, that is antibody populations are added and deleted from the system as becomes necessary. The rate at which this happens is controlled by 4 parameters;

1. the extinction threshold, e_t
2. the extinction phase length e_p
3. the stagnation threshold ϕ_t.
4. the stagnation phase length ϕ_p

If the fitness of the immune system composed of the best member of each population following reproduction and evaluation of each population does not increase by at least ϕ_t over ϕ_p generations, then a new population is added to the system, with randomly generated members. Similarly, if the best member of a population does not recognise at least e_t antigens from the current antigen population, and the population has been in existence for at least e_p generations then that population is removed from the system. A limit of M populations (and therefore M antibodies in the final immune system) is imposed on the system to prevent it growing too large (and therefore too specialised). This is similar to the Potter model but with two differences. Firstly, the learning phase parameter has been added in order to give each population an opportunity to evolve. This is particularly important in a non-static environment. Secondly, the operation of the extinction threshold is modified so that the best antibody in a population must recognise a minimum number of antigens with the caveat that if an antibody recognises an antigen that *no other* antibody from another population recognises, then the population is allowed to continue existing. In the original model described in [Potter and De Jong, 2000], an antibody from a population must contribute a minimum proportion of the total fitness of the serum in order to survive. However, as this quantity cannot be easily isolated in the SDM/immune model, we have modified the approach.

5.2.3 The Evolutionary Algorithm

The EA controlling evolution of each population is identical to that described in [Potter and De Jong, 2000], in order that a fair comparison can be performed. Each population is of size 50, two-point crossover is applied at a rate of 0.6, and a bit-flipping mutation operator used at a rate equal to the reciprocal of the chromosome length. Generational reproduction is applied, using fitness-proportionate selection based on scaled fitness.

5.3 Overview of Experimental Setup

Three series of experiments were designed to investigate the capability and behaviour of the system outlined above. The first series of experiments was designed simply to test the performance of the model on a set of static datasets, in order that its performance could be compared to other published algorithms. The second and third series are concerned with using the model in a non-stationary environment, to see if clusters can be found and tracked in time-varying data. One series of experiments concerns data which varies over time in a random manner. The other is concerned with investigating the performance of the system with datasets in which data appears in cycles, and is designed to test the ability of the system to react more quickly to antigens it has previously been exposed to, and therefore the long term memory of the system.

5.3.1 Default Parameters

Unless stated otherwise, a default set of experimental parameters, given in table 5.1 is used in all experiments. In this table, the parameters marked with at $*$ are taken directly from [Potter and De Jong, 2000]. Others have either been adapted to suit COSDM or are unique to the COSDM model. All experiments are run 10 times, and the mean recall fitness $\bar{r}$ measured at the end of 200 generations of COSDM. The maximum value of recognition radius is set to slightly less than $1/2$ the length of the antigen strings as this has been shown theoretically to be desirable [Kanerva, 1988]. The reasoning behind this as as follows: two random strings of length L are almost certainly $1/2L$

Parameter		Value
Population Size	p	50
Crossover	2-point	$p = 0.6$
Mutation	bit-flip	$p = 1/L$
Stagnation threshold	ϕ_t	0.5
Stagnation phase length	ϕ_p	10
Extinction threshold	e_t	5 antigens
Extinction phase length	e_p	10 generations
Maximum number of population	M_{max}	10
Minimum number of population	M_{min}	2
Maximum recognition radius	R	31
Length of antigen	L	64
Length of antibody	L	64

Table 5.1: COSDM fixed parameters

apart, for large L. If the radius is equal to $1/2L$, then too many strings will fall within each centre. If the radius is much less than $1/2L$, then almost nothing will fall within each centre, thus the radius should be just under $1/2L$.

5.3.2 Comparison of results

The algorithm of Potter *et. al* was re-implemented according to the details described in [Potter and De Jong, 2000]. This algorithm is referred to as CE-POTTER from here on. The results given in [Potter and De Jong, 2000] for experiments on static data-sets containing half, quarter and eighth schema were verified, and the re-implemented algorithm was then used to repeat experiments performed with COSDM so that results could be compared.

A second method of comparison used as a benchmark for the COSDM are the results that would be obtained using the best possible single string generalist in each experiment. This is to confirm that the strategy of locating multiple niches in the data is indeed effective (regardless of how those niches are located). In each of the datasets

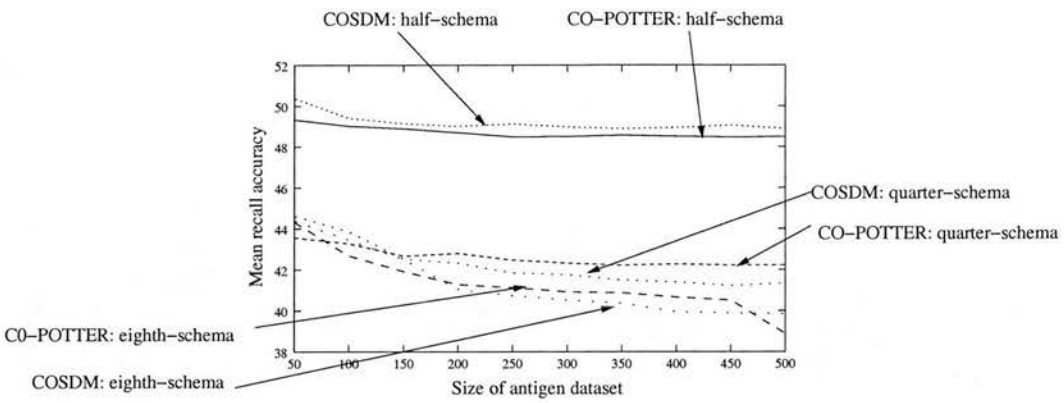

Figure 5.5: Comparison of performance of COSDM and CE-POTTER on static datasets generated from half-schema, quarter-schema and eighth-schema

used in the following experiments, the best possible string generalist would consist of a string in which all bits were set to 1. This string would match at least d bits of each antigen, and on average 50% of the remaining bits, therefore would achieve a mean recall fitness of $(d + \frac{L-d}{2}) = \frac{L+d}{2}$.

5.4 Experiments using Static Data Sets

An initial series of experiments was performed to establish how the new model COSDM performed on static data sets compared to other previously published work. As the number of clusters is known *a priori* in each of these experiments, and we are only interested in whether or not we can discover them, we simply evolve an equal number of populations as clusters in each experiment, i.e both the maximum and minimum number of population parameters are fixed to exactly equal the number of clusters.

Figure 5.5 compares the performance of COSDM to CE-POTTER on datasets generated from half, quarter and eighth schema, containing between 50 and 500 antigens. Although COSDM outperforms CE-POTTER on all datasets generated using half-schema, its performance on the quarter-schema datasets is comparatively worse than CE-POTTER when $N > 150$ and when $N > 200$ on eighth-schema datasets. In all

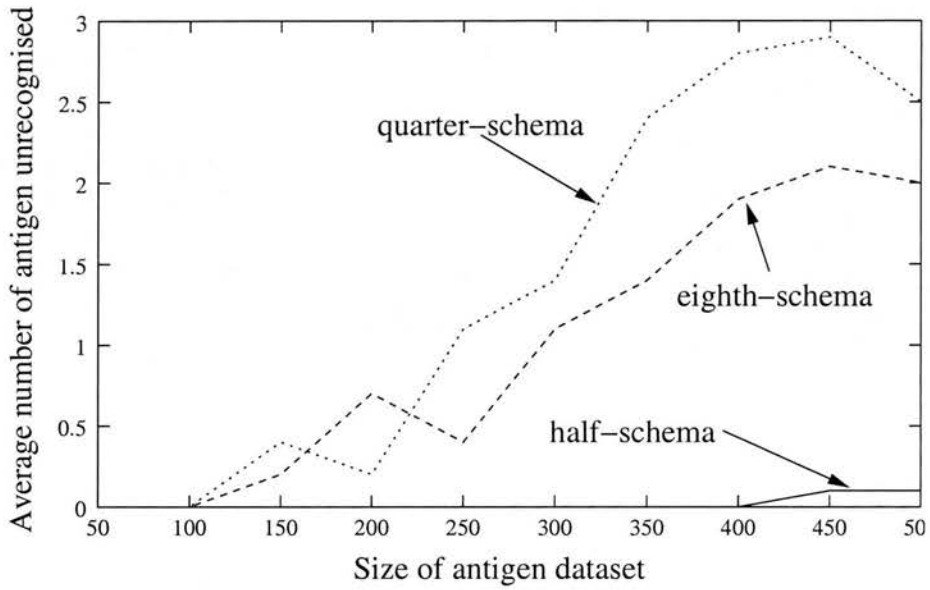

Figure 5.6: The figure shows the average number of antigens in a dataset that are not recognised by the best immune system evolved using COSDM for each of the three categories of datasets

cases however, the performance of COSDM exceeds that of the optimal string generalist. Some insight into this is gained from examining the average number of antigens that are *not* recognised by the best immune system evolved during COSDM for each value of N — these results are shown in figure 5.6. For datasets generated from half-schema antigens, the evolved immune system fails to recognise antigens only when N is large, and then only occasionally. The situation is different for those datasets generated from quarter-schema and eighth-schema; in both cases, from $N > 100$ there appears to be a steady increase in the number of antigens not recognised, somewhat surprisingly this is more exaggerated in the quarter-schema experiments. When the defined schema-length d is small compared to the length L of the schema, there is a higher probability of matching the random part of the schema and hence it is easier to locate suitable hard locations. Note however, that in all experiments, the maximum percentage of antigens remaining unmatched is less than 1% of the total size of the dataset which is likely to be trivial in a real-world database.

Finally, the evolved radii of the hard locations are examined for each category of datasets; for all half-schema experiments, the average radius of the antibodies evolved by COSDM is either 29 or 30, with no obvious trend with increasing N. The average radii of *all* antibodies evolved for quarter-schemas for all values of N is 30, and for eighth-schema experiments, the evolved average radius varies (again with no obvious pattern) between 28 and 29. Therefore, there is a tendency in all experiments for the radius to tend towards the maximum possible, as this allows more antigens to be recognised. This finding is confirmed by repeating the experiments but allowing 6 bits to define the radius, giving a maximum possible radius of $(2^6 - 1) = 63$. For the eighth-schema experiments, no change is observed, for all values of n, i.e. the number of antigens in the dataset, the radius varies between 28 and 30, and the number of unrecognised antigen follows the same pattern observed in figure 5.6. However, for the quarter-schema experiments, for values of $n \geq 350$ the radius rises well above the theoretical maximum upto a value of 45 for $n = 500$ — this is coupled however with a corresponding *decrease* in the number of antigens that are not recognised by the evolved immune system for all n. Similarly, for half-schema, the evolved radii increase with increasing n to reach a maximum of 47 at $n = 500$, and in this case, no antigen are unrecognised, even at $n = 500$.

Recall that the mean recall accuracy quantity is an average over all antigens in the dataset, and therefore antigens which are not recognised by the immune system have a recall accuracy of zero, and therefore have a large effect on the mean accuracy. Clearly this results in a trade-off between recognising as many antigens as possible vs the accuracy with which antigens can be recalled. In the case of the eighth-schema experiments, increasing the number of antigens recognised does not lead to an increase in the mean accuracy of recall, therefore the radii evolve towards the maximum predicted by the theory. For quarter-schema datasets however, increasing the number of antigens recognised produces an increase in mean recall accuracy — even though the antigens are recalled less accurately than compared to the case when the maximum value radius is set to 31, the fact that more antigens can be recognised and hence stored in the memory compensates for this. The same arguments apply to the half-schema datasets in which of course a randomly generated string would match on average half of the bits

in each antigen. In conclusion, it would appears better to fix the maximum radius to be similar to the theoretical maximum of 1/2 the length of the string to enable clusters defined by a small recognition radius to be recognised accurately.

In summary, COSDM appears to perform satisfactorily in all experiments performed in that it does locate clusters within the data in each case, when compared to the optimal single string generalist. Its performance degrades as the size of the dataset increases when COSDM is compared to the model published by Potter *et. al.* This is probably due to the inability of the model to locate all antigens in the dataset — there is no pressure exerted by the fitness function to steer the location of antibodies towards parts of the input space containing unrecognised antigens, therefore it is conceivable that some antigens remain un-noticed by the algorithm throughout the entire evolution. This could be addressed in a number of possible ways. One option would be to include some form of penalty function in the fitness function, based on the number of unrecognised antigens, however this approach has well documented difficulties ([Smith and Coit, 1997]). Also, it should be noted that the COSDM model was run with exactly the same parameters reported by Potter *et al*, and a search of the COSDM parameter space was not performed. As the parameters published in [Potter and De Jong, 2000] are likely to be the result of an extensive optimisation process, it is possible that performing such an optimisation with COSDM may improve its performance.

However on balance, it was felt that the performance of the model was sufficiently promising that it should be tested in a non-stationary environment, where it has already been suggested that an immune system metaphor should provide some advantages.

5.5 Experiments Using Random Non-Stationary Datasets

This section describes experiments performed on random non-stationary data generated using the method described in section 4.2.2. In all experiments, 100 antigens are generated from schema of length $L = 64$. Experiments examined the effect on the mean recall fitness $\bar{r}$ of the number of schemas used to generate the data set, s, the

Number of Schemas	$s \in (2,5,10)$
Defined Length	$d \in (8,16,32)$
Update Rate	$U = 50$
Number of Schemas replaced	$1 \leq g \leq s$

Table 5.2: Values of parameters tested in random pattern tracking experiments

length of the defined section d, the update-rate U and the number of schemas replaced at each update g. Values tested are given in table 5.2. Each experiment was repeated 5 times — $\bar{r}$ was recorded following each of 200 generations. In each group of 5 experiments, the schemas were always generated from the same seed, so that the values of $\bar{r}$ could be averaged meaningfully. For information purposes the average overlap of the defined sections of the entire schema set in each case was calculated; this is defined as the average number of bits that are equivalent *and* defined (i.e. set to 1) in any 2 schemas. Thus, for example, schemas `111###` and `#111##` have an average defined overlap of 2. These values are given in table 5.3. As expected, as the number of defined bits increases, the average defined overlap between schemas dramatically increases. A large defined overlap implies that there will be a large overlap between clusters, with the consequence that changes in the data should be easier to detect, as the antibodies within the immune system do not have to move to new parts of the input space in order to recognise new data.

As the aim of the following experiments is to assess the ability of an adapted COSDM to track patterns in non-stationary data, the system is allowed to undergo a *tolerization period* in which it learns to cluster the patterns in the initial dataset. The ability of the system to then react to *changes* in the data can then be measured from this starting point (rather than from time 0 in which the system contains random antibody data). The length of the tolerization phase in all experiments is set to 200 generations, and recording of results begins from this point.

Number of schemas s	Defined Length d	Average Defined Overlap
2	8	1.06
2	16	4.09
2	32	20.61
5	8	1.11
5	16	4.92
5	32	21.22
10	8	1.06
10	16	4.76
10	32	21.55

Table 5.3: The table shows the average defined overlap between the entire schema set for each set of experiments performed using random non-stationary data

5.5.1 Results

Figure 5.7 shows example outputs from experiments in which $s = 5$, and $d = 8, 32$, and $g = 1, 5$. The graphs illustrate the overall trends observed in *all* experiments: at each antigen update following the tolerization period, there is a rapid drop in $\bar{r}$, immediately followed by a rise in fitness during the next 50 generation period over which the maximum fitness achieved attains a similar level to that observed before the change. Higher values of $\bar{r}$ are achieved when $g << s$ in all cases, but when d is small (and hence there is very little defined overlap between schemas), there is little observable difference as g varies.

Further analysis of the results shows an approximately linear relationship between the magnitude of the drop in fitness following each antigen update and the number of schemas defining antigens that are replaced for all values of s and d tested. Figure 5.8 shows representative results obtained when $d = 16$ and $s = 5, 10$. As the number of antigens replaced increases, it is clear that the system is less likely to be able to cluster the new data, especially as $g \rightarrow s$. Note however, that even in the extreme case when

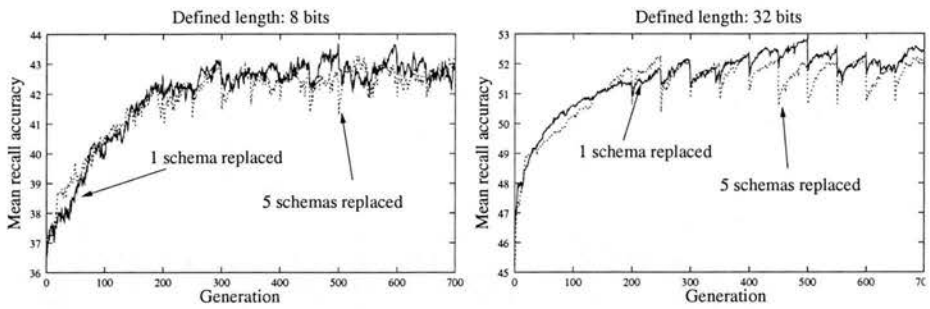

Figure 5.7: Output from experiments in which datasets generated from 5 schema were updated at intervals of 50 generations by replacing g schema

$g = s$, the resulting value of $\overline{r}$ immediately following the antigen update is still very much greater that the fitness observed at generation 0 when the system is randomly initialised. A clearer indication of the success of the model in adapting to the new data is shown by examining the difference Δ between the maximum value achieved at the end of the tolerization period $\overline{r}(U_0)$ (generation t=200) and the maximum value of $\overline{r}$ achieved following each subsequent update, $\overline{r}(U_i)$, i.e. $\Delta = \overline{r}(U_0) - \overline{r}(U_i)$. Figure 5.9 illustrates representative results, showing the value of Δ for the extreme values of g in experiments in which $d = 16$ and $s = 5, 10$. Although the values of Δ fluctuate above and below the zero-line, the magnitude of the deviations is very small compared to the actual value of $\overline{r}$ achieved at the end of the comparatively long tolerization period:- the maximum negative deviation is 3.43% of $\overline{r}(U_0)$, and the maximum positive deviation 5.29% of $\overline{r}(U_0)$ across all experiments. Therefore we tentatively conclude that the COSDM model exhibits some ability to track moving datasets.

5.6 Experiments using Cycling Non-Stationary Datasets

Earlier discussion of the properties of the biological immune system suggested that the basis of learning in such systems is attributable to its long-term memory capacity, which enables it to respond more rapidly and more effectively to subsequent encounters with antigens. Therefore, it is essential to investigate whether the proposed arti-

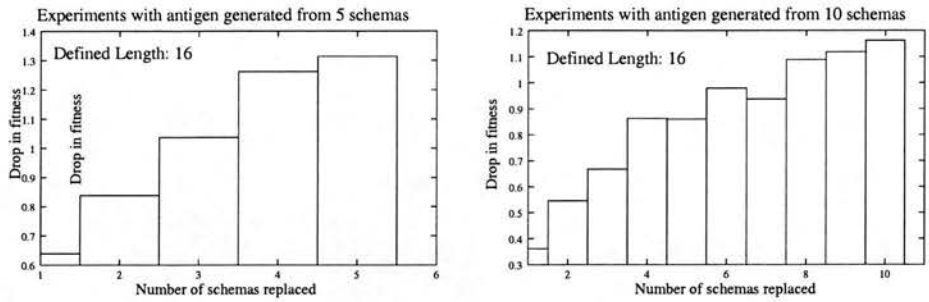

Figure 5.8: The figure shows the magnitude of the drop in fitness following each antigen update for experiments in which $d = 16$, and $s = 5, 10$

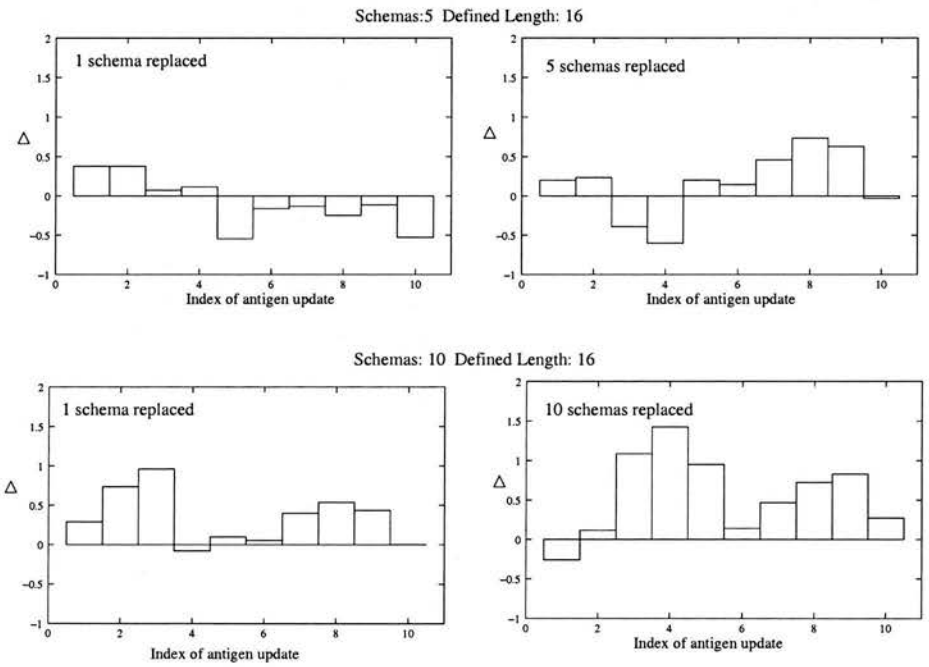

Figure 5.9: Δ vs index of antigen change for experiments in which $d = 16$, and $s = 5, 10$. The extreme cases of $g = 1$ and $g = s$ are shown in each case

ficial model COSDM exhibits any form of memory retention. This is investigated by examining the performance of the model in a modified non-stationary environment in which data is presented to the system in cycles — this enables complete clusters to be re-introduced to the system at regular intervals, and the response time of COSDM to these re-entrant clusters can be measured.

5.6.1 Data Generation

Antigen data is generated using a modified version of the generic method for generating non-stationary data presented in section 4.2.2.

In step 1 of this algorithm, $k * s$ schema are initially generated at random. At any time t, only s of these schema are used to generate the antigen population. A sliding window of size s defines which schemas are used; this window is moved w schemas along the schema list every U generations. The schema list is treated as cyclic and wraps around when the window reaches the end. Thus, if $k = 2$ and $s = 4$, then 8 schemas are initially generated, for example labelled 0,1,2,3,4,5,6,7. If w is equal to s, then all antigens are replaced at each update; thus at time $t = 0$, antigens $\{0,1,2,3\}$ define the data set. At time U, antigens $\{4,5,6,7\}$ define the data, at time $2U$, antigens $\{0,1,2,3\}$ again define the data etc. A more incremental update is achieved by setting $w < s$. In this manner when a cluster is re-introduced, it is not necessarily in conjunction with the same set of other clusters to which the model was originally exposed.

5.6.2 Experimental Results

Experiments were performed in which $s = w$, where $s \in (2,5,10)$, and k initially set to 2. The update parameter U was set to 50, and all schemas were of length $L = 64$ with $d = 8$ defined bits. All experiments displayed similar trends; we present results here for a representative case in which $s = 5$ and $k = 2$. Therefore, at every update, the entire antigen set is replaced, and the schema set defining those antigens alternates between two possible sets, set A defined by schemas 0,1,2,3,4 and set B, defined by schemas 5,6,7,8,9. Figure 5.10 shows this results of this experiment, averaged over 5 different runs. The performance of COSDM is compared to an equivalent experiment using

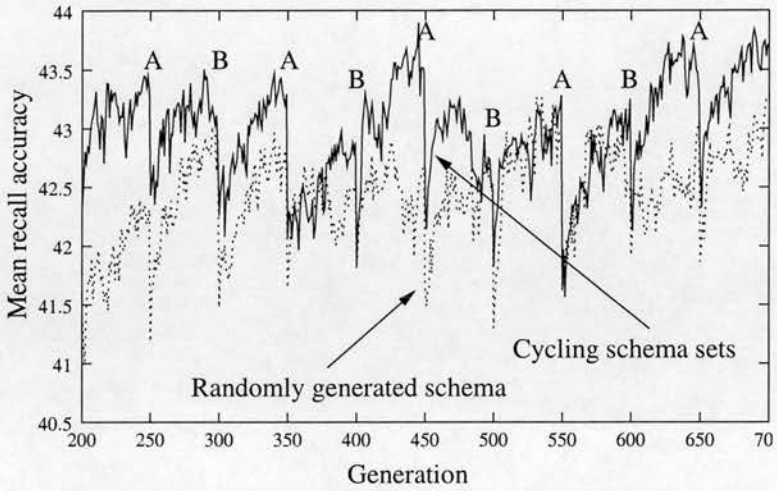

Figure 5.10: The figure compares the performance of COSDM on datasets generated from random schemas, to datasets generated from two schema sets A, B which are re-introduced at regular intervals. The datasets in each case are generated from 5 schema, each schema containing 8 defined bits.

COSDM in which the entire antigen set is updated from randomly generated schema at each update, rather than the alternating schema set just described. The figure clearly shows that COSDM applied to the cycling data set produces better results than when applied to random moving data sets, suggesting that some kind of memory effect is being observed. (As in previous experiments a tolerization period of 200 generations was applied in order to allow the system to learn the initial dataset.)

In order to investigate the *period* of this memory, i.e. the length of the intervals between re-introduction of familiar clusters, we repeated the above experiments setting k to 3, and then 4. We then examined the best fitness achieved for schema set A in each case during every interval for which COSDM was exposed to this schema set. These values were averaged, and the mean and standard deviation calculated for each experiment defined by (s,k). The results are given in table 5.4.

There is little observable difference between the mean values of $\overline{r}$ obtained as k varies, and Students t-tests applied with 95% confidence limits show no significant differences. Therefore, it appears that the COSDM model is able to exhibit some form of memory for past clusters, and for the values of k tried, the extent of this memory is

Number of schemas	Value of k		
s	2	3	4
2	44.63 (0.186)	44.37 (0.274)	44.77 (0.322)
5	43.06 (0.382)	44.21 (0.342)	43.01 (0.405)
10	42.72 (0.266)	42.78 (0.474)	42.79 (0.281)

Table 5.4: The mean and standard deviation (shown in brackets) of the maximum value of $\bar{r}$ found for schema set A, for varying combinations of s, the number of schemas from which the dataset is generated, and k, the multiplier producing the overall schema set

not affected by the size of the intervals between reappearance of clusters.

5.7 Conclusions

A new algorithm, COSDM, has been introduced which uses a genetic algorithm to evolve the position and radii of the hard locations in a sparse distributed memory. It was postulated at the beginning of this chapter that this would allow clusters in large, binary datasets to be identified. A co-evolutionary architecture was used, which it was hoped would provide a means of automatically determining the number of clusters in the data, when this is unknown *a priori*. Furthermore, the architecture was designed to enable the algorithm to be used in a non-stationary environment, so that it could theoretically track moving clusters.

The experiments described show that the algorithm shows limited success. First consider its performance in static environments (although the arguments now presented apply equally to the non-static experiments also) — the experiments detailed in section 5.4 show that although datasets containing 2 clusters are clustered more accurately than when using the CE-POTTER algorithm, performance of COSDM degrades as the number of clusters and the size of the datasets increases. Figure 5.6 may explain this; COSDM fails to recognise increasing numbers of antigens as the size of the dataset size and number of clusters increases. The fitness function, i.e. $\bar{r}$ rewards memories which on average are able to recall data more accurately; however, there is no driv-

ing force within this function to encourage *exploration* of the space of hard location positions. If an item of data is not recognised by the initial randomly generated hard locations, then there is no pressure, other than random chance, to drive the position of the antibodies towards those unrecognised antigens. Therefore, although the genetic algorithm fine-tunes the positioning of those antibodies so that data *initially* recognised is more accurately recalled as the algorithm progresses, some items of data remain unrecognised throughout the algorithm. This effect obviously becomes more apparent as the size of the datasets, and the number of distinct clusters increases, as the probability that data will not be recognised by randomly generated antibodies increases. The fact that the recognition radii tend to converge towards the maximum allowable value reinforces this idea — increasing the radii increases the likelihood that an antigen will be recognised, and therefore its recalled accuracy will contribute to $\bar{r}$. The mechanisms for population creation and deletion provide a possible means for increasing the coverage of the antigen space, although currently population are created only in response to the value of the fitness of the system becoming static, hence for the reasons outlined above, this does not drive the system to cover unrepresented parts of the space, other than by chance.

Potter *et al.*'s algorithm, CE-POTTER, does not suffer from this effect, as their fitness function involves a direct *competition* between the best member of each population; in each competition, there will always be at least one winner, no matter how poor the recognition between antigen and antibody. Contrast this with COSDM, in which not only is the ultimately recalled antigen a result of *cooperation* between the best member of each population, but some antigens may not be recognised at all, due to the values of the recognition radii. Finding suitable recognition radii is key; even if evolution were to prove capable of determining the correct values, the method described requires that a maximum value be placed on each radius. Theory described by [Kanerva, 1988] shows that it is inadvisable to set the radii to greater than half the length of the strings, otherwise the probability of each antibody recognising every antigen is too high. On the other hand, although small radii are desirable for accurate clustering, so that each antibody only recognises a single cluster, then the positions of the antibodies become crucial; if they are incorrectly placed, then the majority of

data will be unrecognised. (Recall that the original postulates of the SDM specify that the storage locations or hard antibodies are randomly distributed in the $\{0,1\}^n$ address space and that they are given from the start; clearly neither postulate applies in this context.)

There are a number of ways in which the problem of unrecognised antigens could be addressed within the constraints of the proposed architecture, many of which rely on the introduction of some kind of penalty function, as adopted in many applications of evolutionary algorithms to constrained problems. It is noted from the outset however that there is a wealth of EA literature documenting the difficulties of penalty based approaches, for example [Richardson et al., 1989], and opinion still varies on whether the approach is justified.

Aside from the issue of unrecognised antigens, COSDM is slower than the CE-POTTER algorithm, which would become a more serious issue as the size of the databases to which it is applied increases. CE-POTTER requires calculating the match-score between a member of each population and each antigen at every fitness evaluation; on the other hand, COSDM requires calculating each match-score as in CE-POTTER to determine if the antigen lies within the recognition radius of the antibody, then storing the antigen at the antibody if so, and hence updating the counters, and then finally performing recall of the entire data set, during which counters must be summed for each antibody recognising the antigen to identify the recalled antigen, which then has to be compared to the original antigen. Therefore, each fitness evaluation is a time consuming process.

Despite these flaws, the results of applying COSDM to a non-stationary problem environment are promising; the algorithm does exhibit the capability to track moving data, and also exhibits a basic form of memory. These are precisely the properties of the immune system metaphor that we hoped to encapsulate. This implies that the SDM component of the model may be of value; however, in the model as it stands, the potential value of the SDM approach is somewhat outweighed by the difficulties associated with using an EA to evolve the positions and radii of the antibodys. It has already been alluded to that choosing the right value for the recognition radii is crucial — this is a fundamental feature of the original SDM. Therefore, it appeared that in

order to progress this work, two features of the COSDM model needed to be addressed, namely that of the engine by which the correct locations of the hard antibodies could be found, and a method to circumvent the difficulties associated with the recognition radii. The next chapter presents a new model — SOSDM — which borrow from a modifed version of Kanerva's SDM suggested by [Hely et al., 1997] and disregards the EA in favour of a self-organising metaphor more in keeping with the principles of the immune system we are attempting to mimic.

Chapter 6

A Self-Organising SDM — SOSDM

6.1 Introduction

The previous chapter concluded that the combination of an EA with an immune system metaphor suffered from three major drawbacks; namely, that the evolved immune systems failed to recognise some antigen altogether, evolving the correct recognition radii for each antibody was extremely difficult, and that the system was relatively slow to evolve, owing to the nature of the fitness function. In this chapter, another method of discovering an SDM immune system capable of adapting to non-stationary data is described. The new model, self-organising SDM or SOSDM, relies on an important principle of the biological immune system not yet explored in this thesis — its self-organising nature.

As previously discussed, the original form of the SDM is essentially a static memory, with fixed hard locations. COSDM represented hard locations by antibodies, and attempted to evolve the best definition of those antibodies, subject to the current state of the environment using an evolutionary algorithm. However, SOSDM views the memory as a truly self-organising system. Initially randomly placed antibodies self-organise in order that they become distributed throughout the input antigen data space in a manner which reflects the input antigen data distribution. This seems an entirely logical step — the immune system itself is self-organising, whilst viewed from the computational angle, there is an abundance of

literature describing algorithms for self-organising systems. Furthermore, a number of data-clustering algorithms rely on self-organising principles, ([Jain et al., 1999] for a review), and also as noted in the literature review in chapter 2, attempts have been made at applying network-based immune system models to data clustering [De Castro and Von Zuben, 2000b, Timmis and Neal, 2001].

This chapter briefly reviews the basic principles of self-organising systems, and describes an alternative model of an SDM, before describing how these two approaches can be combined to produce a model capable of rapidly and efficiently clustering data. Experiments are performed on the same datasets used to analyse COSDM, so that a direct comparison of the two systems can be made.

6.2 A Brief Background on Self-Organising Maps (SOMs)

The SOM in various forms has commonly been used to visualize and interpret large high-dimensional data sets as well as to perform clustering. The earliest example of a SOM was proposed by Kohonen, [Kohonen, 1982a] and is commonly used for clustering purposes. Typically, a map consists of a number of units or neurons between which there is a specific topological relationship. The map or network is trained by an iterative procedure in which the units in the network are gradually adjusted to reflect the clustering of the training data. The training procedure arranges the network so that units representing centres close together in the input space are also situated close together on the topological map. The basic learning algorithm follows a two step procedure which is iterated over many epochs:

1. Determine the unit that is closest to the input data, 'the winner'
2. Adjust the winning unit *and* its neighbours to be more like the input case, using a weighted sum of the input case and the unit itself

A time-decaying learning rate is applied to the adjustment to ensure that as time progresses, the updates become more subtle, and the map eventually stabilises into a

representation of the input data. The topological ordering is achieved by also updating units in the neighbourhood of the winner. The size of the neighbourhood also decreases over time, so that as the map becomes fine-tuned, the size of the neighbourhood eventually becomes zero and only the winner is updated.

In the typical Kohonen network, the number of units and topology of the network is predefined. However, the choice of network structure is difficult, and the need to define a decay schedule for the various features is problematic [Fritzke, 1997b]. A typical network produced by the Kohonen algorithm is shown in figure 6.1, taken directly from [Fritzke, 1997b]). This illustrates the difficulty of choosing a suitable topology that matches the underlying data distribution. Furthermore, the use of a decay schedule is one of the key reasons why this type of simple self-organising map has generally been considered unsuitable for handling non-stationary data distributions. The decaying network adaption parameter means that as the value of the parameter approaches zero, the network becomes static and therefore cannot react to any further changes in data.

Another self-organising system that can distribute units according to some given probability distribution is the *neural gas* algorithm of Martinetz and Schulten, [Martinetz and Schulten, 1991]. This algorithm is capable of distributing centres to reflect underlying data distributions, but does not provide any topological information, therefore there are no connections between units in the distribution. The algorithm works by determining the distance in input space between the units in the network and an input signal, and then adapting the units based on the *rank order* of these distances. Like the Kohonen networks, it requires a decay schedule to be defined in advance for the adaptation parameters, and the number of units must also be pre-defined. However, it is better able to adapt to data distributions as it does not provide topological information. Figure 6.2 again taken from [Fritzke, 1997b] illustrates the application of the Neural Gas algorithm to the same data distribution as shown in figure 6.1.

In order to overcome some of the problems associated with Kohonen type networks that have topological structure, Fritzke has proposed two algorithms for incrementally growing SOMs: growing cell structures, [Fritzke, 1994], and the Growing Neural Gas (GNG), [Fritzke, 1995]. Both these models provide topological information as well as distributing the centres according to the underlying distribution. The

models present an incremental method of growing a network, in which there is no need to specify the size of the network, and in which all parameters are constant. These models therefore are much more suited to finding clusters in data about which no decisions can be made *a priori* regarding the likely number of clusters and suitable topologies. In the GNG algorithm, a growth mechanism for incrementally growing a network is combined with the topology generation of competive Hebbian learning [Martinetz and Schulten, 1991]. Starting with very few units, units are inserted successively. To determine where to insert such units, local error measures are gathered during the adaptation process, and each new unit is inserted near the unit with the most accumulated error. The complete algorithm is given in figure 6.4. The Growing Cell Structures algorithm is very similar to the GNG model, but differs in that the network topology is constrained to consist of k-dimensional simplices, where k is some positive integer, chosen in advance. Thus $k = 1$ specifies a line, a triange is specified by $k = 2$ etc.

An example of the application of the GNG algorithm is given in figure 6.3. In its original forms however, although the GNG is capable of following slowly changing probability distributions, for example a normal distribution with a slowly drifting mean, it is unable to handle rapid changes in distribution in a non-stationary environment, [Fritzke, 1997b]. The problem arises due to units in the network becoming stuck in former regions of high probability density and from then on becoming 'dead' units, with no connections to any other unit in the map. Fritzke proposes a solution to this in [Fritzke, 1997a] in which he introduces a new on-line criterion for identifying useless neurons on the network. According to [Fritzke, 1997a]:

> ... When this criterion is used in the context of the (formerly developed) growing neural gas network model to guide deletion of units, the resulting method is able to closely track non-stationary distributions. Slow changes of the distribution are handled by adaptation of existing units. Rapid changes are handled by removal of "useless" neurons and subsequent insertion of new units in other places

Each of the above algorithms incorporate features which might potentially be incorporated in an immune-SDM based algorithm for data clustering. Clearly, they are all self-organising, a characteristic which it seems obvious to try and capture in an im-

mune algorithm given the nature of the biological immune system itself. This thesis is not concerned with data clustering with topological information, therefore the neural gas algorithm would seem to provide a basis for a new immune-based self-organising algorithm. However, as mentioned, it has several drawbacks, such as the need to define a decay schedule and to predetermine the number of units. Therefore, we propose that elements of the incremental GNG algorithm might also be incorporated into the new model. Furthermore, the GNG has already been shown to be capable of adapting to non-stationary probability distributions which is encouraging.

6.3 Modifying an SDM to function in a non-stationary environment

In chapter 4 section 4.1.4, the postulates underlying Kanerva's original SDM were outlined, and it was explained why these postulates are unsuitable for modelling an SDM which functions in a non-stationary environment. However, [Hely et al., 1997] have proposed an alternative model of an SDM — although the model was developed in order to handle non-random input data more satisfactorily than Kanerva's original system, it contains several features which could be adapted to work in a non-stationary environment. According to [Hely et al., 1997]

> the SDM signal model retains the essential characteristics of the original SDM whilst providing the memory with a greater scope for plasticity and self-evolution. By removing many of the problematic features of the original SDM the new model is not as dependent upon *a priori* input values.

Their signal-model SDM modifies postulates 1 and 3 of those given in section 4.1.4 and introduces a 4th postulate. Thus the new postulates (taken directly from [Hely et al., 1997], are given below, with the fundamental changes highlighted in italics:

1. *The storage locations that make up the final memory are* not *known from the start. Initially locations are created until there is an excess of storage locations which then compete for available signal. Storage locations receiving little or no*

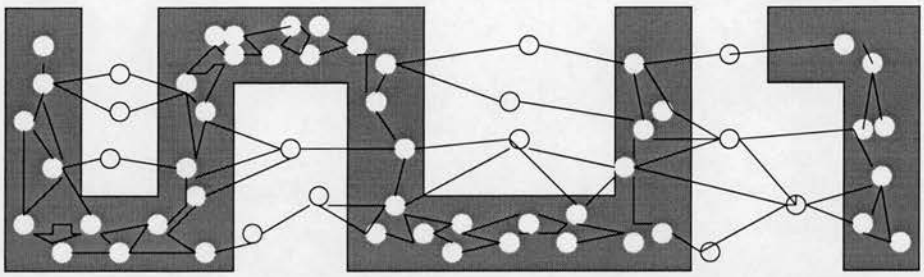

Figure 6.1: Kohonen map result for a clustered distribution which is uniform in the shaded areas. Due to a mismatch between the data distribution and the network topology the data distribution is not well represented.

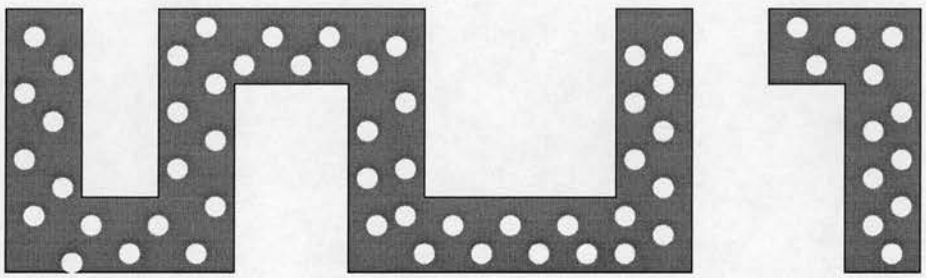

Figure 6.2: Neural Gas result: the distribution of the units reflects the underlying data distribution. There is no topological information, i.e. there are no neighbourhood connections

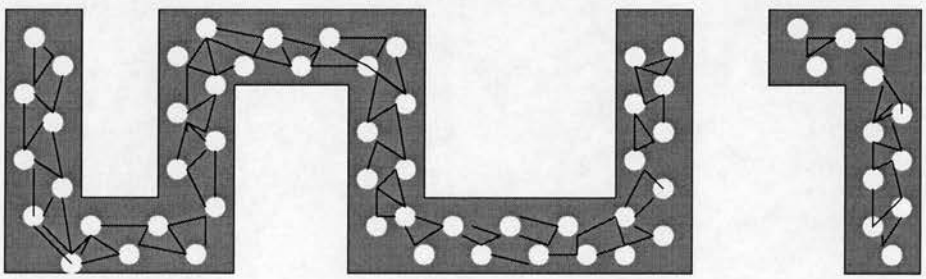

Figure 6.3: Result of applying the Growing Neural Gas algorithm: the topology is very well adapted to the data distribution. The structure consists of two clusters reflecting the clustered data and all units lie in the regions in which the data actually comes from.

1. Initialise the set $\mathcal{A}$ to contain 2 units, c_1 and c_2, with reference vectors chosen randomly according to $p(\xi)$.

   Initialise the connection set $C, C \subset \mathcal{A} \times \mathcal{A}$ to the empty set:

2. Generate at random an input signal ξ according to $p(\xi)$.

3. Determine the winner s_1 and the second nearest unit s_2

4. If a connection between s_1 and s_2 does not exist already, create it, and set the age of the connection between s_1 and s_2 to zero.

5. Add the squared distance between the input signal and the winner to a local error variable:

6. Adapt the reference vectors of the winner and its direct topological neighbours by fractions of the total distance to the input signal.

7. Increment the ages of all edges emanating from s_1

8. Remove the edges with an age greater than some parameter a_{max}. If this results in units having no more emanating edges, then remove them as well.

9. If the number of input signals generated so far is an integer multiple of a parameter λ, insert a new unit near the unit which has accumulated most error

10. Decrease the error variables of all units

11. If a stopping criterion has not been meet (e.g. net size or some performance measure) continue with step 2

Figure 6.4: Fritzke's GNG algorithm

signal are removed. Locations which survive are chosen for the total amount of signal they receive.

2. The storage locations are very few in comparison with 2^n, i.e. the memory is sparse.

3. *Although storage locations are initially randomly distributed in the* $\{0,1\}^n$ *address space, the final distribution of locations depends on the input patterns presented, and may be non-random.*

4. *The recognition radius of the original SDM is replaced by a new parameter which decreases the value of the signal as it spreads out. Locations have real valued counters to store a copy of the data, weighted by the strength of signal they receive. The signal does not propagate after it falls below a minimum strength*

The key attractions of this approach as far as SOSDM is concerned is that the model abandons the recognition radius parameter, which has already been shown to be problematic, and that it does not rely on locations being randomly distributed throughout the input space; clearly, the input data in a database is not random. The need for an explicit recognition radius is removed by distributing each data pattern, i.e. signal, throughout the memory with decreasing strength. The centre closest to the input data receives 100% of the signal; thereafter the signal spreads throughout the memory and some small percentage of the signal, say 5%, is lost at each subsequent location encountered. Each location thus receives a weighted copy of the signal, which is used to update real-valued counters. A 'generate-and-cull' approach is taken to producing the final memory. Initially, a new location is created for every input data pattern, with an address identical to the input data. A 'trial period' then occurs in which a sample sample of the input patterns are written to memory, with locations competing for signal in the manner described. At the end of this period, a 'killing phase' begins. A further sample of patterns is written to the memory, only this time, locations which have received the least signal are killed off at regular intervals until only approximately 1/2 the original number of locations remain. At this point, the counters are reset and the

memory is considered stable. The model was shown to exhibit greatly improved efficiency when presented with non-random address patterns when compared to Kanerva's original SDM.

However, despite these attractions, the signal model has several drawbacks when placed in a non-stationary data-clustering context. Firstly, the 'generate-then-cull' approach is inappropriate, particularly when considering very large datasets. The primary reason is that this method does not lend itself well to an adapting environment — the culling process simply forces the memory to converge onto the current input space. If that input space changes, another cycle of add-then-cull would be required. Furthermore, the existence of distinct phases is not very appealing, as this in itself requires some external mechanism to cause transition between the phases. Secondly, the method of distributing data across the memory, albeit with decreasing strength is unsuitable, especially in a memory that contains relatively few hard locations. In small memories, all centres are likely to receive some proportion of the signal if the parameters are not chosen extremely carefully — this is exactly the opposite of what is desired in a clustering system, in which the intent is to isolate clusters. Finally, there is a large overhead in distributing large amounts of data to many nodes, however, the signal model seems to require a large number of nodes in order to function correctly.

Thus, the new model SOSDM now described borrows from the underlying philosophy of the Hely signal model of distributing data, but modifies the detail somewhat. Antigens in the new model can bind to multiple antibodies, but with a binding affinity based on the attraction between an antibody and an antigen relative to all other antibodies in the system. SOSDM also borrows from the CE-POTTER algorithm in that antibodies compete for antigens based on their *affinity* for the antigen data, i.e. the similarity between the data-item and antibody as given by its address. In order to tackle the problems associated with the signal model's generate-and-cull' approach, SOSDM adopts a similar approach to *growing* an SOM as that taken by Fritzke in his GNG algorithm, in that as the network grows, antibodies are added and deleted only as necessary in areas of the input space that are misrepresented.

It should be noted that there have also been other attempts to address the shortcomings in Kanerva's original model, and the choice of Hely's model as the one to adapt

was because it seemed to lend itself most obviously to an immune-based model. For example [Sjödin, 1996] tried to refine the basic model so that it could more efficiently deal with non-randomly distributed data by adding an extra counter to each location which counts the number of items stored at the location. A further location is added covering the entire space — these are then used to determine which locations should be used in any read attempt from the memory. [Sjödin, 1996] shows that this method greatly reduces errors in the recalled strings for data that is biased when compared to the original model, as the new model ignores many locations which are activated but effectively contain noise.

6.4 Implementation of SOSDM

Pseudo-code outlining the SOSDM algorithm is given in figure 6.5. Firstly, antigens are distributed to a subset of antibodies, based on the *affinity* of each antibody for the antigen in a batch process. *Affinity* is simply the Hamming Distance between an antigen and an antibody - the closer the distance, the stronger the affinity between the two. This results in the counters of the subset of antibodies being updated, according to the strength of each antigen encounter. After all antigens have been given a chance to encounter an antibody, the accumulated error of each antibody is calculated. The error is equivalent to the sum of the *distances* between each antibody and any antigen it recognises, weighted by the strength of the encounter. The value of the error is then used to allow the antibodies to self-organise — antibodies gravitate towards areas of the space in which they recognise data, the distance and direction of the movement determined by the accumulated error. Each of these steps is now described in greater detail.

6.4.1 Notation

The following notation is used to describe the manner in which SOSDM is implemented. Assume an SOSDM is defined by n antibodies (i.e hard locations in an SDM), each of which is referred to as c_i. Each antibody is described by two strings, each of length L. The first, $V(c)$ denotes the address or location of the antibody, the second

1. begin with a fixed number of antibodies N, with randomly initialised positions and counters set to 0.
2. present a subset $s(s \leq N)$ of the data-set (antigens) visible at time t to the SOSDM
3. distribute the data in the s to *each antibody* in the SOSDM, with a *strength* proportional to the *affinity* of the antibody for the data
   - update the counters at each antibody according to the binding affinity of the antigen-antibody encounter
   - compute the accumulated error at each antibody
4. update antibody positions — the distance and direction of the move is determined by the total accumulated error at the antibody
5. update antibody counters
6. add or delete nodes from the memory if necessary
7. go back to step 2

Figure 6.5: The SOSDM algorithm

$C(c)$ consists of L real-valued counters. Thus, $V(c_{ij})$ specifies the address of bit j in antibody i and $C(c_{ij})$ specifies the counter value of bit j in antibody i. The aim of the SOSDM is to cluster a dataset consisting of N binary antigens, a. Each antigen is of length L, and a_i denotes the value of bit i in antigen a.

6.4.2 Distributing the Data

Data is distributed through the SDM according to the *distance* $\mathcal{A}$ of each antibody c from an input antigen a. This is simply defined as the Hamming Distance between the antigen a and the *address* of the antibody c equation 6.1).

$$\mathcal{A}(c_i,a) = \sum_{j=1}^{j=L} \begin{cases} 1 & if \quad V(c_j) = a_j \\ 0 & otherwise \end{cases} \tag{6.1}$$

The affinity of each of the N antibodies for the input antigens is calculated. Following this, the antibody that is closest in distance to the antigen a, denoted by $\mathcal{A}^*$ can be determined:

$$\mathcal{A}^* = max(\mathcal{A}(c_1,a),....,\mathcal{A}(c_N,a)) \tag{6.2}$$

This value $\mathcal{A}^*$ is then used to determine the strength of the antigen-antibody encouter, i.e. the *affinity* of the antibody for the antigen. From an SDM perspective, this value determines how much signal from the input data is distributed to each centre. The affinity of any antibody for an antigen is proportional to the ratio of the distance of the antibody from the antigen compared to the distance of the 'winning' antibody from the antigen, i.e. $\mathcal{A}(c_i,a)$ to $\mathcal{A}^*$. A further parameter known as the affinity-threshold t is introduced, such that $(0 \leq t \leq 1)$. Antigens are only considered to bind to those antibodies in which $\mathcal{S}$ is greater than this threshold. This is shown in equation 6.3. Again, from an SDM perspective, this means that only centres where $\mathcal{S}c,a > t$ have their counters updated due to the incoming signal.

$$\mathcal{S}(c,a) = \begin{cases} \frac{\mathcal{A}}{\mathcal{A}^*} & if \quad \mathcal{S}(c,a) > t \\ 0 & otherwise \end{cases} \tag{6.3}$$

Binding between an antigen and antibody implies updating the counters at that antibody. The counter $C(c_{ij})$ for each bit j at each antibody c_i is updated according to equation 6.4, where γ is equal to 1 if $V(c_{ij}) = 1$ and to -1 if $V(c_{ij}) = 0$.

$$C_(c_{ij}) = C(c_{ij}) + \gamma\mathcal{S}(c,a) \tag{6.4}$$

Each time an antibody binds with an antigen, it increments an internal variable $\mathcal{R}$ which relects the total amount of binding exhibited by the antibody, as shown in equation 6.5:

$$\forall c: \quad \mathcal{R}(c) = \mathcal{R}(c) + \mathcal{S}(c,a) \tag{6.5}$$

6.4.3 Calculating the Error at Each Antibody

The self-organising mechanism by which antibodies move around the immune system is based on a calculation of the total error accumulated at each antibody *after all* antigens have been distributed to the system. Error is calculated in the following manner; firstly, each time an antibody binds to some antigen c, the error *at each of the j bit positions* for the address of that antibody is updated according to equation 6.6. The error at each bit position is thus effectively a measure of the difference between the *desired* value of the antibody address at position j as given by the value of the antigen at position j and the *actual* value of the antibody address, $V(c_{ij})$.

$$\mathcal{E}(c_{ij}) = \mathcal{E}(c_{ij}) + \mathcal{S}(c_i, a)(a_j - V(c_{ij})) \tag{6.6}$$

Movement of antibodies only occurs after all antigens have been presented to the system, which allows the total average error at each antibody, $\overline{\mathcal{E}}$, to be calculated, according to equation 6.7. Note that this will always have a value lying between -1 and 1.

$$\overline{\mathcal{E}}(c_{ij}) = \mathcal{E}(c_{ij})/\mathcal{R}(c_i) \tag{6.7}$$

6.4.4 Updating the nodes position and counters

Once all antigens have been presented, self-organisation of the antibodies can take place. Thus, as shown identified in steps 4 and 5 of the SOSDM algorithm in figure 6.5, the *address* of each antibody is modified as the antibodies move to parts of the input space more representative of the antigens they are binding to. The counters associated with an antibody also move, however they too are modified as the physical locations of the antibodies move to reflect the new position of the antibody.

The probability with which the position *and* the counter of each bit j in an antibody c_i are moved is defined according to the *absolute* value of the average error $\overline{\mathcal{E}}(c_{ij})$. If the value of $|\overline{\mathcal{E}}(c_{ij})|$ is greater than 0.5, then this value determines the probability with which an address bit is flipped and its counter updated. (The introduction of the seemingly arbitrary value of 0.5 ensures that the system will eventually stabilize,

given a static data set, and prevents random movements). Thus, if $\overline{\mathcal{E}}(c_{ij}) < 0$, then $V(c_{ij}) \Rightarrow 0$, and if $\overline{\mathcal{E}}(c_{ij}) > 0$, then $V(c_{ij}) \Rightarrow 1.$. Equation 6.8 summarises the effect on the *counters* for each bit j in each antibody c_i for all antibodies in which $|\overline{\mathcal{E}}(c_{ij})| > 0.5$. A new parameter is introduced — the *influence-counter*, I. This parameter allows the amount by which the counters are adjusted to be explicitly controlled.

$$C(c_{ij}) \Rightarrow C(c_{ij}) \times \left(1 + \left(I \times \overline{\mathcal{E}}(c_{ij})\right)\right) \tag{6.8}$$

Thus, the effect on a counter is that it is increased or decreased by a percentage of its original value, the amount of which is proportional to the total error accumulated by the antibody. The effect on the address of bit j is that it is flipped, with a probability proportional to the average error accumulated at that address location.

In summary, the key features of the SOSDM system involve distributing a sample of antigen-data to the system, followed by allowing the system to self-organise, in a manner dependent on the average error accumulated by each antibody. The algorithm is iterated until it stabilises (given a static set of antigens). Note that when using SOSDM there is no need to calculate the mean recall accuracy of the system at each iteration, unlike with COSDM. The value of this parameter does not feedback into the algorithm and has no bearing on its performance. However, in order for the observer to evaluate the performance of SOSDM, this quantity must be calculated. The method by which this is done is now outlined.

6.4.5 Recalling Data from the SOSDM

Exactly as with COSDM, the quality of the SOSDM defined by this model is measured by the accuracy with which data, i.e. antigens, stored in the memory can subsequently be recalled, i.e. by $\bar{r}$. The recall mechanism is almost identical to that already described for COSDM in section 5.2.1.2, chapter 5. To recap, when attempting to recall an antigen a, first the antigen that is retrieved from the memory a' is calculated, and then this is compared to the desired antigen, i.e. that which was originally stored in the memory, a. The process is as follows:

- Calculate the subset of antibodies n' for which the binding affinity $\mathcal{S}(c_i, a) > t$

- Sum the counters of each member of the subset n' at each of the j bit positions to give $\sigma_j(a)$. The value of each counter $C(c_{ij})$ is weighted by the binding affinity $\mathcal{S}(c_i, a)$ during the summation process, as shown in equation 6.9.

$$\sigma_j = \sum_{i \in n'} C(c_{ij}, a) \times \mathcal{S}(c_i, a) \tag{6.9}$$

Thus, the only differences between this method and that used to measure recall in COSDM are that the subset n' is derived from those antibodies in which the binding affinity $\mathcal{S}(c_i, a) > t$, rather than requiring the use of a recognition radius, and that during the summation process, the counter values are weighted. From here on, the recall process proceeds exactly as for COSDM, i.e. according to equations 5.3 to 5.5. Thus, the actual recalled antigen is calculated, compared to the desired antigen, and the match-score $\mathcal{M}$ between the actual and desired antigen derived. The average of the match-scores over the entire antigen set is used to calculate $\bar{r}$.

6.5 Calibrating the SOSDM

This section describes a series of experiments that were performed in order to test and calibrate the new model. Comparisons are performed to the CE-POTTER algorithm, as in chapter 5. Extensive testing was performed in order to determine the bounds in which the model performs satisfactorily, and also the ease with which it could be calibrated. In the form outlined above, the model requires only 2 parameters to be set, which compares favourably to any evolutionary algorithm experiments and also to the number of parameters that must be be set in many of the competitive learning algorithms described above. Those parameters are the binding affinity threshold, t, and the influence-counter, I.

6.5.1 Experimental Set-up

Experiments described in the following sections are performed using the static datasets outlined in chapter 4, section 4.2.1. Therefore, experiments are performed on datasets

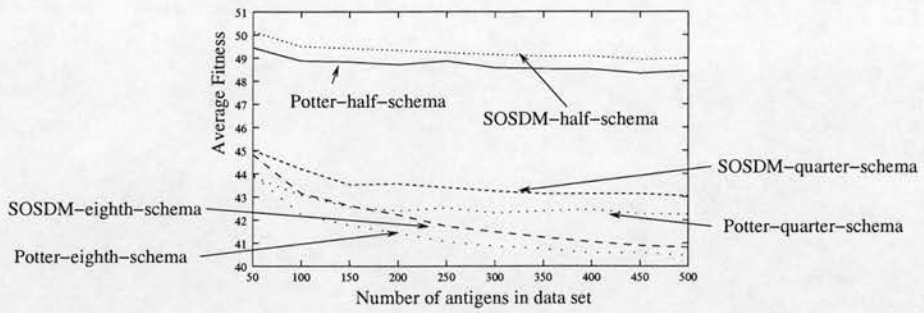

Figure 6.6: Comparison of Potter Algorithm to SOSDM for all experiments

containing 2,4, and 8 clusters, identified as half-schemas, quarter-schemas, and eighth-schemas respectively. The number of antigens in each dataset is varied from 5 to 500, in steps of 50, and the length of each antigen string is always 64. Unless stated otherwise, each experiment is repeated 10 times, and the SOSDM algorithm is applied for 200 iterations. The quality of the immune system representing the data is measured by the mean recalled accuracy, (see equation 5.5), as in the COSDM experiments. As with COSDM and CE-POTTER, the number of antibodies in each experiment was fixed before the experiment began, and remained static throughout each experiment, as the number of clusters in each dataset is known *a priori*.

6.5.2 Comparison of SOSDM Performance to that of CE-POTTER

Initial experiments were performed with $t = 1.0$ and $I = 1.0$. Thus, antigens can bind to *all* antibodies with $\mathcal{A} = \mathcal{A}^*$ and to no others. (This is in direct contrast to the Potter approach in which an antigen can bind to only a single antigen, with ties broken by age of antibody). The setting for I also ensures that counters are adjusted maximally. The best recall-accuracy obtained in each of 10 experiments is recorded, and the results averaged. Figure 6.6 shows a plot of the results — clearly SOSDM outperforms CE-POTTER for all sizes of antigen datasets and regardless of the number of clusters. T-tests show that the mean recalled accuracy obtained using SOSDM is statistically significant in every case when compared to the identical experiment using CE-POTTER — these results are tabulated in appendix B.

The results may be partially explained by examining the number of antigens that

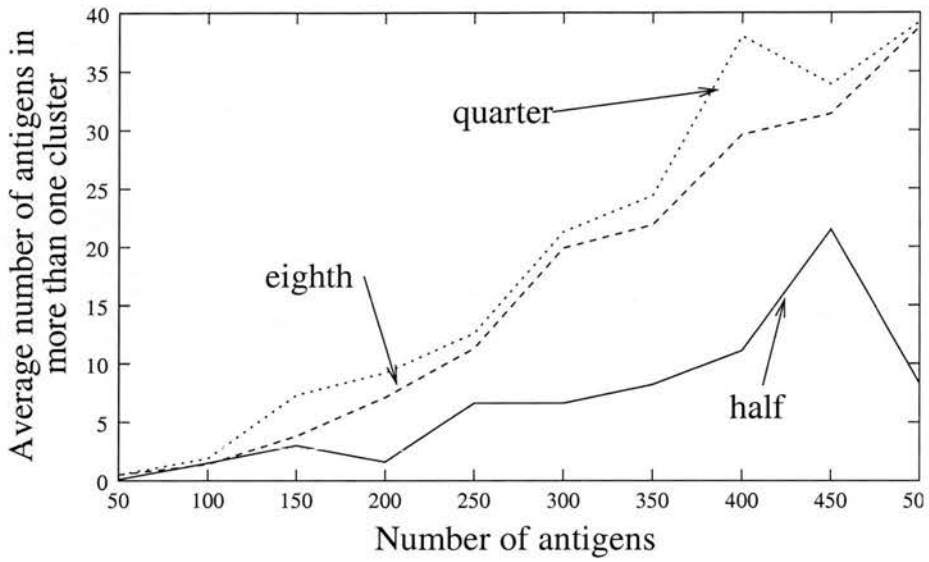

Figure 6.7: Examining the number of antigens that bind to more than one antibody

bind to more than one antibody, i.e. belong to more than one cluster — this is shown graphically in figure 6.7. This shows that as the number of antigens increases, the number of antibodies binding to an antigen increases. These findings apply to all experiments. In small datasets, it is relatively straightforward for the antibodies to distinguish between each cluster. For very large datasets however, even though the antigens nominally belong to separate clusters, there is likely to be a large overlap between items in each cluster, especially as the length of the defined section characterising each cluster decreases, and the number of antigens generated from that schema increases. Thus, the memory must generalise in order to accurately recall the large number of data-items, despite the fact that items nominally belong to a finite set of clusters — this is achieved by allowing clusters to overlap. This effect is much more clearly apparent in the quarter-schema and eighth-schema than it is for those using half-schema.

6.5.3 Number of Iterations Required to Find the Best Solution

Table 6.1 shows the mean number of iterations required to produce the best solution for all experiments, with the corresponding standard deviations. We observe that the mean values show that the algorithm rapidly converges on a solution, however the algorithm

Antigens	Half		Quarter		Eighth	
	Mean	SD	Mean	SD	Mean	SD
50	15.400000	14.683324	39.700000	52.679429	46.200000	46.322061
100	14.700000	5.618422	37.000000	24.805913	42.600000	41.679198
150	16.700000	11.671904	42.500000	30.613178	75.500000	37.146108
200	27.600000	26.854340	57.200000	30.017773	103.500000	46.980492
250	13.300000	9.866329	46.800000	46.499223	98.800000	47.377444
300	23.700000	18.481522	71.400000	48.339540	110.500000	48.376418
350	13.500000	6.023104	64.700000	56.330473	107.500000	52.816769
400	16.800000	14.226735	79.100000	53.831940	144.500000	36.939139
450	10.400000	8.248906	53.100000	57.085608	148.700000	53.804275
500	18.100000	13.714955	46.700000	40.260402	89.100000	47.799233

Table 6.1: SOSDM: Average/SD of epochs taken to find best solution

is somewhat variable — the standard deviations are very large. Closer examination of the results showed that in the majority of runs, a good solution was found in very few iterations, but occasionally, a run required a large number of iterations, thus resulting in the large standard deviation, Even if this is taken into account, these results compare very favourably with those obtained by Potter — each iteration of SOSDM requires at most $n * c$ calculations of match-score. Potter's algorithm on the other hand requires $n * c * p$ where p is the population size controlling each species (or antibody) per iteration, plus the usual overheads associated with reproduction in an evolutionary algorithm (for example, crossover, mutation etc.).

6.5.4 Investigating the sensitivity of SOSDM to the *influence-counter* parameter

Initial experiments fixed I, the influence-counter parameter to 1.0. This section describes the effect of varying this parameter. All previous experiments were repeated, varying I in each case from 0.0 to 1.0 in steps of 0.1. Again the mean recalled accuracy was taken as the measure of quality of the result. Surprisingly, the value of I appeared to have little effect on the final result. Students t-test was applied to the mean recalled-accuracy of all possible pairs of experiments (x, y) in each possible ex-

Number of Antigen	Type of Schema		
	half	quarter	eighth
50	7	2	4
100	4	1	0
150	5	0	5
200	8	3	4
250	3	6	5
300	0	0	2
350	0	12	0
400	0	3	0
450	0	1	2
500	2	2	4

Table 6.2: The table shows the number of comparisons (out of a total of 55 in each case) that gave statistically significant differences between the mean recalled accuracy as I was varied

periment class (A,C) where A = number of antigens, and C = number of clusters, and x and y identify the value of the influence parameter (with the caveat that $x \neq y$). Table 6.2 shows the number of comparisons which gave statistically significant results, in that the probability that the means are different is ≥ 0.95. Clearly the table shows that very few comparisons of I gave statistically significant results. (The total number of comparisons per experiment class (A,C) is 55). In cases where a statistically significant difference was observed, then the higher value of I gave improved results, therefore in future experiments it was determined that I should simply be fixed at 1.0.

6.5.5 Choosing the Binding Affinity Threshold, t

As described in section 6.4.2, the binding affinity threshold t determines if an antigen can bind to an antibody, and if so, how well. High values of t introduce more competition and therefore encourage antibodies to specialise, whereas low values will encourage overlap of clusters. Note that a randomly chosen antibody c and antigen a

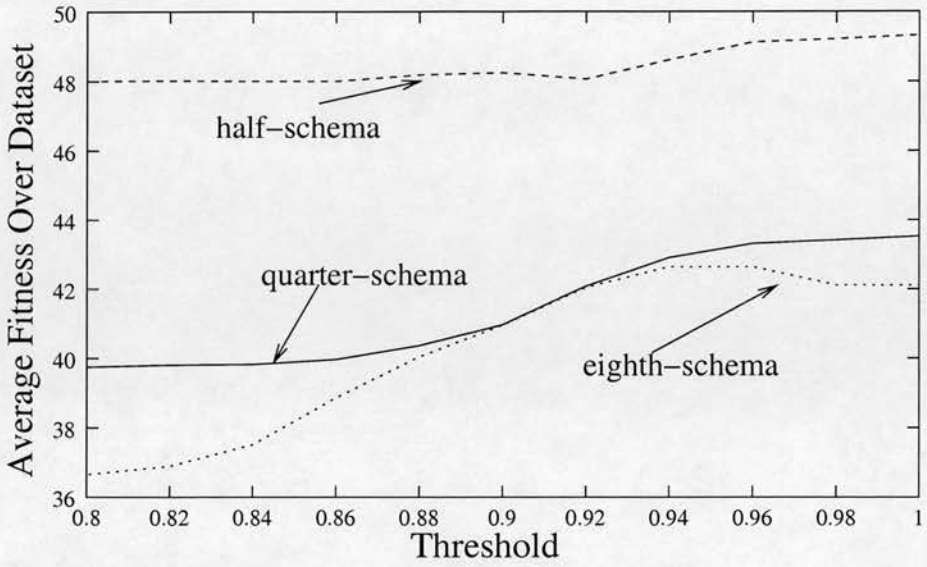

Figure 6.8: Examining effect of threshold parameter t on average best fitness across data set

will be expected to have a binding affinity of $\mathcal{S}(c,a) \geq 0.5$, as in a binary system on average 1/2 of the data bits will match the antibody bits, therefore we need only consider thresholds above this value. In practice, it is observed that much higher values are required if data is not to be distributed to every antibody.

Experiments compared the best recalled accuracy for values of t ranging from 0.8 to 1.0 in steps of 0.02 for 3 datasets. Each dataset contained 200 antigens; set 1 was generated from half-schemas, set 2 from quarter-schemas and set 3 from eighth-schemas. Figure 6.8 shows the results of these experiments: as expected, recall-accuracy increases with increasing t, as the ability of the system to generalise is reduced. All 3 experiments show a band over which recall-accuracy rises rapidly, before flattening off. As a rule-of-thumb, a threshold of $t \geq 0.95$ seems a sensible choice.

6.6 Limitations of the Model

This section investigates the limitations of the SOSDM model in terms of its scalability and with respect to the characteristics of the data-sets it should cluster. In order for it to

prove ultimately useful using real-world data, the model should perform successfully with very large data-sets, and also with very long antigens. The latter requirement is particularly true if we are considering encoding data that occurs naturally in a non-binary form as a binary antigen string. A database may easily contain one hundred attributes per record, and encoding a single one of those attributes may require many bits; encoding a 5 digit US zip-code for example would require 16 bits. Furthermore, data-sets will vary in both the number of clusters they contain, and the distribution of data within those clusters. Thus, this section describes experiments which consider the above factors.

6.6.1 Investigating the effect of cluster size

In the system outlined so far, every antigen will bind to at least one antibody at each iteration of the algorithm, regardless of the affinity $\mathcal{A}$ of the data for the antibodies, i.e. there must always be an $\mathcal{A}^*$ for each antigen. If all clusters in the dataset contain roughly equal numbers of items then the recall rate for each cluster will tend to be roughly similar. However, in real datasets this is unlikely to be the case — clusters will be unequal in size, and some may be very small compared to others. In this case, clusters containing very few items will tend to be 'swallowed' into other clusters as the antibodies are pulled towards the larger clusters. This will be reflected in a low accuracy of recall for items in small clusters. This is verified by generating a series of new datasets in which the clusters are generated in unequal proportions, and repeating some of the above experiments using both SOSDM and CE-POTTER.

Data is generated by modifying the algorithm described in section 4.2.1 of chapter 4 as follows:

- The total size of each dataset is fixed at 200 antigens
- For each experiment, n clusters are represented by n schemas, each with defined length d, and the remaining $L-d$ bits represented by wildcards.
- The size of cluster 1 is set to x

- The size of the other $(n-1)$ clusters is set to be $(200-x)/(n-1)$, i.e. antigens are equally distributed to each of the remaining clusters.
- Antigens are generated from each schema according to the size of the cluster by replacing the wild-cards in the schema at random.

Using this method, antigens are generated from half-schema, varying x from 10-100 in steps of 10, from quarter-schema by varying x from 10-50 in steps of 5, and from eighth-schema by varying x from 2-25 in steps of 2. 10 experiments were repeated for each dataset, using both SOSDM and CE-Potter. The best recalled-accuracy for the entire dataset and for cluster 1 was averaged over the 10 experiments in each case. The results are shown in figures 6.9, 6.10, and 6.11. An identical trend is observed in both SOSDM and CE-Potter results — whilst the recall-accuracy of the entire dataset changes little with the size of cluster 1, there is a noticeable increase in recall-accuracy of cluster 1 as the size of the cluster approaches that of the other clusters. Moreover, the accuracy of recall of cluster 1 is often less than the optimal single-string generalist, described in section 5.3.2. For all values of n, (i.e total number of clusters), the result obtained by SOSDM intersects this baseline at smaller values of x (i.e. the number of antigens in the smallest cluster) however. For the half-schema experiments, the recall-rate is below the baseline until the cluster contains approximately 14% of the total antigens. The figure drops to 6% for the quarter-schema experiments, whereas in the case of the eighth-schema experiments, the recalled-accuracy of cluster 1 is always above the baseline, though it shows the same increase as x increases.

6.6.2 Fitness Proportionate Selection of Data — FPS

In attempt to improve the recall accuracy of small clusters, a new method of selecting data to be stored in the memory was proposed. In the original SOSDM, *all* antigens visible to the SOSDM are stored exactly once in the memory at each iteration of the algorithm. An alternative approach is to select antigens for storing in the memory at iteration i according to the *difficulty* with which the data can was recalled from the system at iteration $i-1$. This method is identical to the *fitness proportionate selection*

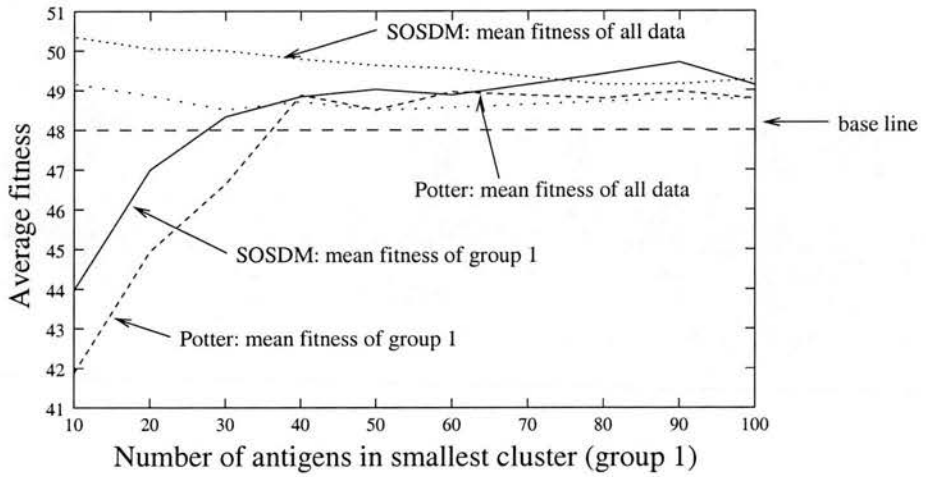

Figure 6.9: Half-Schema: Figure shows average best fitness recorded across entire dataset, and for smallest cluster only. Comparison is shown for SOSDM/Potter algorithms

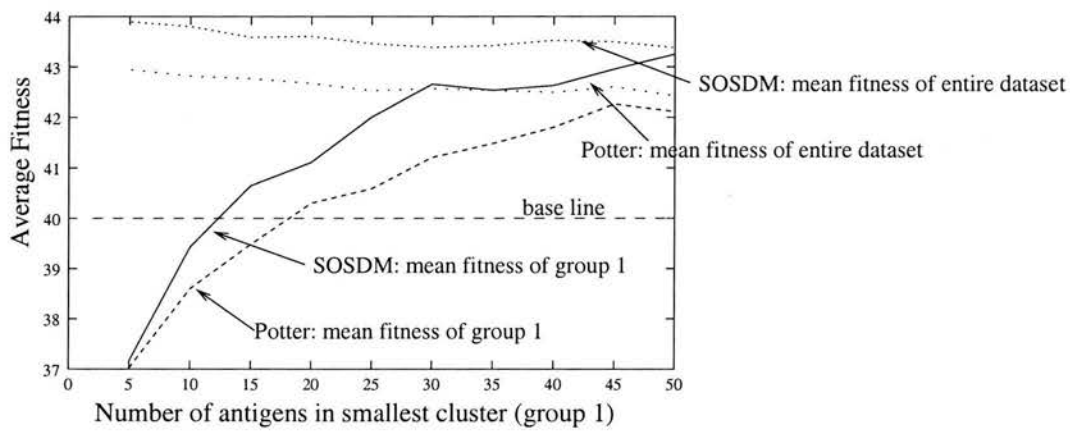

Figure 6.10: Quarter-Schema: Figure shows average best fitness recorded across entire dataset, and for smallest cluster only. Comparison is shown for SOSDM/Potter algorithms

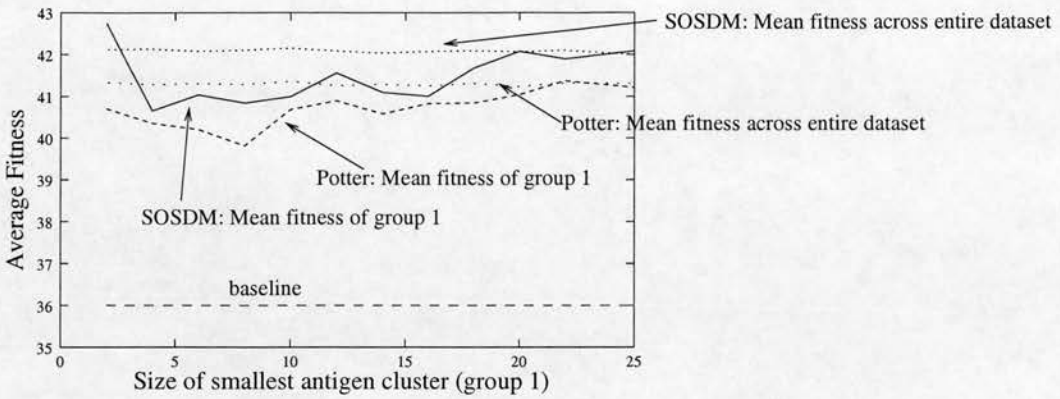

Figure 6.11: Eighth-Schema: Figure shows average best fitness recorded across entire dataset, and for smallest cluster only. Comparison is shown for SOSDM/Potter algorithms

method common to evolutionary algorithms in which items are selected according to some measure of fitness:

- Let f_i be the fitness value of an individual i and let $\bar{f}$ be the average fitness of the population i.e. $\bar{f} = (1/N)\sum_{1}^{N} f_i$
- The probability of an individual being selected is given by:
  $p_i = f_i / \sum_{1}^{N} f_i = (1/N)(f_i/\bar{f})$, since $\sum_{1}^{N} f_i = N\bar{f}$

.

In this case, the *fitness* of a piece of data is inversely proportional to the accuracy with which the data is currently recalled. Therefore items which are recalled poorly have a high fitness and therefore more chance of being selected. A consequence of this method is that the same piece of data may be selected multiple times for storing in the memory during one iteration of the algorithm.

In order to implement this method of data selection, the fitness or *failure rate* f_a for recall of each antigen a at iteration t is calculated as shown in equation 6.10 during the recall phase [1].

[1] At iteration 0 of the algorithm when all counters are 0, the failure rate is defined by $f_a = (L - \mathcal{M}(a))/L$

	CE-POTTER	Standard SOSDM	SOSDM with FPS	
			s=10	s=200
Mean recall all data	49.17	50.28	49.82	50.09
Mean recall group 1	41.88	44.24	46.19	45.33

Table 6.3: Comparison of mean recalled accuracy for CE-POTTER, standard SOSDM and FPS-SOSDM for group 1 schemas and entire dataset

$$f_a(t) = (L - r(a))/L \tag{6.10}$$

Thus, failure rates calculated following iteration t are used at iteration $(t+1)$ in order to select a subset of data of size s for storage.

6.6.2.1 Experimental Results

A series of experiments was performed using a dataset containing 200 antigens generated from two half-schema. 5% of the dataset was generated using schema-1, and the remaining 95% using schema-2. The size of subset s selected for storing in the memory at each iteration was varied from 10 to 200 in steps of 10. After 200 iterations, the best recalled-accuracy of the entire dataset was measured, and also the best recalled accuracy of the 5% of antigen generated from schema-1. These results were compared to the corresponding experiments where all data is stored exactly once in the SOSDM. In each case, experiments were repeated 100 times. Table 6.3 compares the results for CE-POTTER, with standard-SOSDM and FPS-SOSDM.

6.6.2.1.1 Average recall of entire dataset Comparing the mean recall-accuracy for FPS against the standard SOSDM algorithm shows the standard algorithm always outperforms FPS, and applying Student's t-test to the results shows that the results are statistically significant, (see table B.4 in appendix B. The results obtained using FPS are still better than those obtained by CE-POTTER however, and improve as s increases.

Size of subset 1	Size of subset 2	P(means different)	Mean Subset 1	Mean Subset 2
10	20	0.970369	46.193636	45.450909
10	90	0.960757	46.193636	45.539091
10	100	0.993885	46.193636	45.297273
10	110	0.991606	46.193636	45.340909
10	120	0.951153	46.193636	45.570000
10	140	0.975882	46.193636	45.458182
10	180	0.967632	46.193636	45.498182
10	200	0.993710	46.193636	45.333636
50	100	0.959255	45.926364	45.297273
50	200	0.955828	45.926364	45.333636
70	100	0.973447	46.004545	45.297273
70	110	0.964623	46.004545	45.340909
70	200	0.971447	46.004545	45.333636

Table 6.4: The table shows the comparisons of subset size used in FPS which produced a statistically significant difference in mean fitness for the smallest cluster

6.6.2.1.2 Average recall of smallest cluster Examining the mean recall of the smaller group of schemas however shows the opposite trend; for all values of s the best result found is better than both standard SOSDM and CE-POTTER. The differences are statistically significant in each case (see table B.5 in appendix B). Thus, there is a trade-off between improving fitness of small groups vs fitness of entire data set. There is no obvious trend in the results as s increases. In comparisons of mean recalled accuracy of the smallest cluster for a subset size of size s_1 to a subset of size s_2, then in only 13/190 cases are there significant differences observed in the recalled values. The cases where statistical differences were observed are summarised in table 6.4.

6.6.3 Performance of SOSDM vs Size of Dataset

Experiments were performed using datasets ranging in size from 500 antigen to 10,000 antigen in steps of 500. All datasets were generated using 4 quarter-schema, and the mean recalled accuracy of the entire dataset measured at the end of 200 iterations of

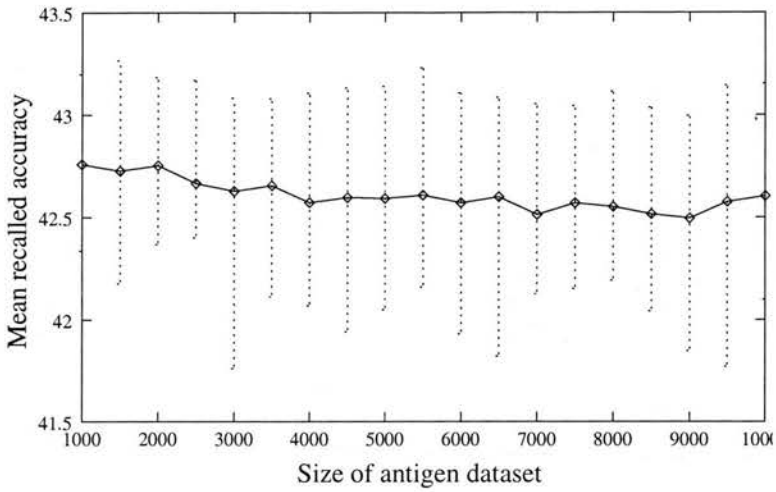

Figure 6.12: The figure shows how mean recalled accuracy $\overline{r}$ varies with the size of the antigen dataset N

the algorithm. Experiments were repeated 50 times in each case. Figure 6.12 shows the performance of SOSDM vs the size of the dataset, with error-bars showing the minimum and maximum accuracy over the 50 experiments. Note that although there is a slight downwards trend in mean recalled accuracy $\overline{r}$, the value of $\overline{r}$ is always significantly greater than the result that would be obtained using the best possible string generalist, which would give $\overline{r} = 40$. T-tests show what there is a significant difference ($p > 0.99$) in the value of $\overline{r}$ obtained for $N = 1000$ and that obtained when $N = 10,000$.

6.6.4 Performance vs Length of Antigen

A second series of experiments used datasets generated again from quarter-schema, this time of fixed size N=200 antigens. The length of the antigen L in each dataset was varied from 40 to 1000 in steps of 40. The best recalled accuracy $\overline{r}$ was measured at the end of 200 iterations of SOSDM, and the results averaged over 50 trials. Figure 6.13 shows the results of these experiments; a comparison is made to the mean recall accuracy that would be expected using the best possible single string generalist for each value of L. The figure shows a direct correspondence between $\overline{r}$ and L — again, for every value of L , the value of $\overline{r}$ exceeds that expected using the single string generalist

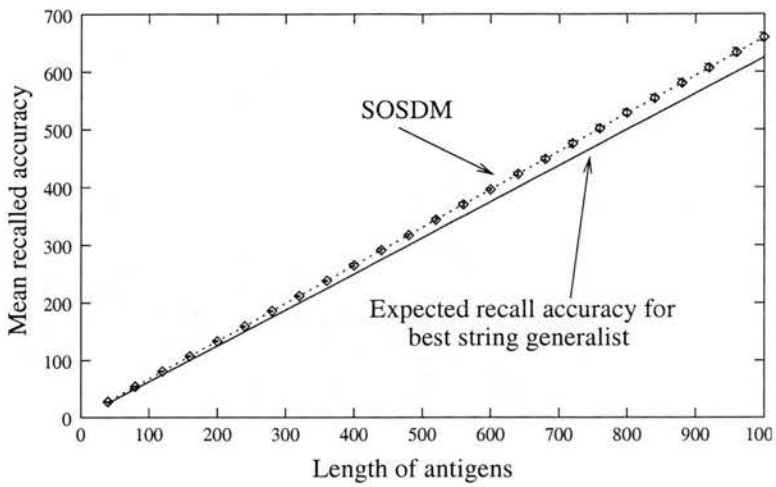

Figure 6.13: The figure shows how mean recalled accuracy $\overline{r}$ varies with the length of antigen L in a dataset of fixed size 200

and this difference increases as L increases.

6.7 Performance of SOSDM in non-stationary environments

One of the motivations that drove the design of the SOSDM system was that it should be able to operate in a non-stationary environment, taking inspiration from the biological immune system. Thus, consider the behaviour of the system in an artificial non-stationary environment in which the number of clusters remains fixed and known *a priori*, but in which the centres of the clusters may move and in which the data within the clusters is continuously updated. This environment allows many types of dynamic scenarios to be tested — two possibilities are addressed here:

1. The centre of the clusters remain fixed, but the data in the clusters is gradually updated, by replacing the original data-items with new data-items, so that the total number of data-items remains constant.

2. Clusters are randomly replaced by entirely new clusters containing new data.

Again the total number of data-items in the system remains constant throughout. (This is an identical scenario to that described in chapter 5 section 5.5).

Experimental details and results are now presented for the behaviour of the SOSDM system under each of these scenarios. In both scenarios, an update regime is implemented in the following manner:

- The first update (whether of an entire cluster or of data-items within a cluster) is made after a fixed interval known as the tolerization period. This allows the system sufficient time to organise into s state where it accurately represent the initial dataset.
- Subsequent updates are performed whenever the fitness of the system has returned to within 1% of the maximum fitness recorded immediately prior to the last update.

The size of the interval between updates (i.e. the number of iterations) indicates how long it takes the system to respond to the change in data and return to its previous level of accuracy, therefore is a useful indicator of the performance of the system. Thus in each experiment, the interval between updates is recorded, and averaged over 5 repeated runs of each experiment.

6.7.1 Update of data within fixed clusters

The experimental procedure adopted is at each update to choose at random *one* of the clusters in the system, and replace a randomly chosen fraction of the antigens within that cluster with new data, which are generated from the schema that identifies the cluster. The number of antigens replaced at each update is controlled by the variable f, which specifies a percentage of the antigens to be replaced. Experiments are performed using an antigen data-set containing 200 items, each of length 64 bits. The antigen are generated from 2 distinct schema, each with 32 contiguous bits set to '1', the remaining bits being wild-cards. The length of the tolerization period is set to 100 iterations, and updates are performed whenever the system returns to within 0.5% of its previous best fitness. The relationship between the length of the update interval and the fraction of

antigens updated, f, is investigated by varying f from 10% to 100% in steps of 10, and allowing the experiment to run for a total of 2000 iterations in each case. The averaged results are given in table 6.5. The results show that when the description of the clusters remains unchanged, on average the SOSDM system quickly responds to new data. The standard deviations are rather large however, as can be seen from the wide variation in the minimum and maximum number of iterations between each update shown in this table. Occasionally, SOSDM takes a very large number of intervals to return to within 0.5% of its previous level of accuracy, though these cases are rare. There is a general (and expected) trend that the number of intervals between updates increases as f increases. However, the average interval size is still very small, even when 100% of the dataset is replaced. It is noteworthy that table 6.5 also shows that in every experiment, the minimum interval required for the system to return to its previous level of fitness is only 1 iteration. This is quite surprising, and simply appears due to a fortuitous combination of the random choice of data to be replaced and the random data with which it is replaced. More detailed examination of the data would be required to confirm this. Nevertheless, it illustrates the model's ability to react rapidly to changes in data.

6.7.2 Appearance of new clusters

A somewhat harder test of the system is to investigate how it responds to the appearance of entirely new clusters. The number of antibodies in the SOSDM model is fixed throughout each experiment, as in this case the number of clusters is of course known *a priori*. Although this is of course an artificial situation, in some ways it represents an extreme test of the system — in real life, entire new clusters are unlikely to suddenly appear at the same time as other clusters suddenly disappear, rather, a more gradual process would occur. The situation in the real world is likely to fall somewhere in between the behaviour described here and that described in the previous section, therefore the extremes are investigated.

For consistency, experiments were repeated exactly as in chapter 5, section 5.5. Thus 9 sets of experiments were performed, in which data sets were generated using $2, 5, 10$ schemas of length 64 bits, and the defined section of each schema was set to

f (%)	Average Interval between Updates	Minimum Interval	Maximum Interval	Standard Deviation
10	1.000000	1	1	0.000000
20	1.126316	1	13	1.224677
30	4.833333	1	252	27.442566
40	3.820225	1	130	15.437851
50	30.024390	1	549	81.543790
60	19.873563	1	340	64.113012
70	32.823529	1	401	82.663375
80	29.536232	1	336	75.658956
90	24.122449	1	280	55.245143
100	33.694915	1	296	74.696959

Table 6.5: The table shows the average number of iterations taken for the SOSDM system to reach its peak fitness value following replacement of f antigens. An *interval* records the number of iterations between antigen updates.

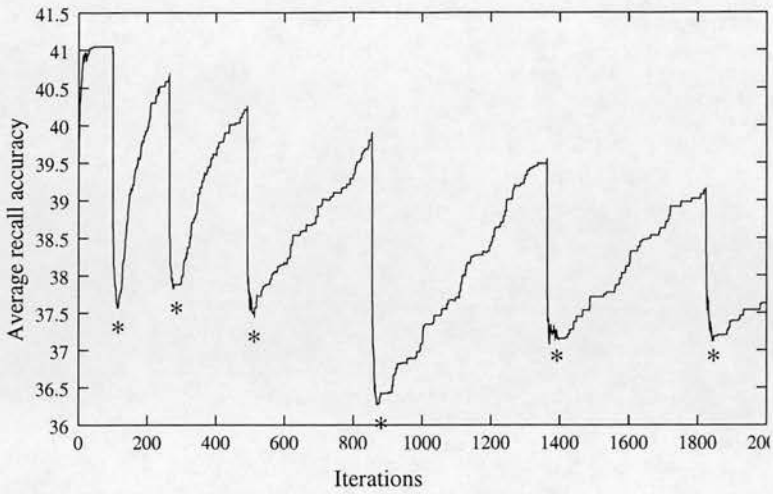

Figure 6.14: The figure shows how the recall accuracy of the SOSDM changes following a number of antigen updates in which entirely new clusters are introduced. The points marked * indicate the iteration at which the antigens were updated. The data contains 5 clusters, each containing 40 antigens. One cluster is replaced at each update.

either 8,16 or 32 bits. For each dataset containing c clusters, experiments tested the ability of SOSDM to respond to replacing $1, 2, .., c$ clusters at each update, resulting in a total of 51 experiments. Figure 6.14 shows a typical result of one of the experiments in which the dataset was generated from 5 schemas each with 8 defined bits, and in which one cluster was replaced at each update. Full results are given in appendix B, tables B.6, B.7, B.8. It is difficult to observe clear trends in the results by varying either the number of clusters in the dataset or the number of defined bits in a cluster. However, it is possible to observe that in all but one of the experiments, the average time lag for the system to return to its previous best level of fitness increases as the number of updates in one experiment increases.

It becomes increasingly difficult for the system to respond as the number of clusters being replaced increases. This is clearly shown by analysing two situations in which an identical number of *antigens* are replaced at each update, but in differing numbers of *clusters*. Thus, if we compare an experiment on a data-set generated from 2 schema in which 1 cluster is replaced with one in which the data-set is generated from 10 schemas and 5 clusters are replaced, (table 6.6), then it is clear that when the number of defined

Number of Defined Bits	Index of Update	Average Lag		Minimum Lag		Maximum Lag	
		S=2	S=10	S=2	S=10	S=2	S=10
8	2	159.2	*251.8*	100	*183*	216	*365*
	3	276.6	*613.4*	187	*309*	364	*1304*
	4	407.0	*758.75*	280	*574*	497	*941*
	5	611.0	657.0*	372	*657*	846	*657*
16	2	100.0	401.0*	100	*401*	100	*401*
	3	138.2	1244.0*	100	*1244*	231	*1244*
	4	128.2	**	100	**	165	**
	5	268.6	**	143	**	348	**
32	2	146.8	*187.0*	100	*132*	206	*257*
	3	324.0	*361.4*	201	*307*	504	*480*
	4	121.6	*300.4*	100	*248*	208	*362*
	5	615.25	*307.0*	459	*198*	819	*522*

* indicates that only *one* update was observed during 2000 iterations

** indicates that *no* updates were observed during 2000 iterations

Table 6.6: The table compares the average lag between updates observed when 50% of the antigen data is updated at each update for data sets in which $S = 2$ and $S = 10$. However, when $S = 2$, this corresponds to replacing 1 cluster, when $S = 10$ this corresponds to replacing 5 clusters.

bits in each schema is 8 or 16, it is much more difficult for SOSDM to respond to replacing 5 clusters. Indeed, in some experiments, the SOSDM never manages to return to within 1% of its previous best fitness before the limit of 2000 iterations is reached, and hence only one update is performed. Furthermore, the tables in appendix B shows that there is little observable difference in the average lag between updates between experiments in which 1 out of a possible 10 clusters (and therefore 10% of total data) is updated and those in which 1 out of 2 clusters (and therefore 50% of total data) is replaced. Therefore, we can conclude that SOSDM seems much more sensitive to changes in cluster position than changes in data.

6.7.3 Making the System Truly Dynamic

In order for SOSDM to operate in a truly unsupervised manner in a non-stationary environment, SOSDM should be able to create and delete antibodies in response to the data it is exposed to, as generally the number of antibodies required will not be known *a priori*. [Fritzke, 1994] notes that a purely incremental approach to generating a network or model is unsatisfactory in a dynamic environment, as the centres (or antibodies in the SOSDM case) must be able to adapt to changing data, and that therefore, a model must also contain a mechanism for removing anibodies when appropriate. Possible ways of achieving this are now considered.

6.7.3.1 Adding Antibodies

The incremental algorithms proposed by Fritzke (e.g. [Fritzke, 1994, Fritzke, 1995] handles addition of nodes by simply adding new units after presentation of every λ input signals. This is unsatisfactory for SOSDM as we wish the number of antibodies to correlate with the number of clusters, hence using such an addition mechanism would work only if combined with an efficient method of removing antibodies. Furthermore, the choice of λ is also difficult.

The method used in COSDM, (chapter 5), was to add antibodies whenever the global fitness of the system had stagnated. This approach seems reasonable for COSDM where the fitness measure of the system has a direct influence on the evolution of the co-evolving populations of potential antibodies; in SOSDM on the other hand, the global fitness or recall accuracy of the system is incidental — it does not have any direct influence on the movement of antibodies during the self-organisation of the system. Therefore, this method appears somewhat of a 'cheat' and again incurs high overheads in having to calculate recall accuracy at every iteration.

Therefore, a mechanism is suggested in which stagnation of the system is detected not in respect to recall accuracy but in terms of movement of antibodies — if no movement of any antibody has happened over a fixed number of generations s (the stagnation threshold) then an antibody is added. The new antibody is generated in a random position with its counters initialised to zeros.

6.7.3.2 Deletion

[Fritzke, 1997a] suggests a method of deleting 'dead units' when trying to track non-stationary distributions using the GNG algorithm that involves a local utility measure. In the typical GNG application, it is possible to compute how much the error for some given input signal ξ would increase if the winning unit s_1 were not present, and the signal instead had to be mapped to the runner-up unit, s_2. The increase in error is then simply $\|\xi - w_{s_2}\|^2 - \|\xi - w_{s_1}\|^2$. This allows the total utility of each unit to be calculated by summing the utilities over all input signals, and then a unit is removed whenever its utility falls below some predefined threshold. However, this method does not transfer well to SOSDM; in the pure GNG algorithm, error is easily calculated as $\|\xi - w_{s_2}\|^2$, but in *SOSDM*, although the winning antibodies are determined based on the correlation between the *address* of the antibody and the data to be stored, the error in the recall accuracy depends also on the *counters* stored at the physical address. Secondly, the recall error can only be calculated once all data has been stored, and hence there is an extremely high overhead in calculating such a utility measure, especially in a very large database. For these reasons, the use of a utility function was rejected as a method of deleting nodes.

The following method of deleting antibodies is suggested :- the sum of the binding affinities of the antibody with all its binding antigens, $\mathcal{R}$ is compared to the total binding affinity the antibody would exhibit if it had bound to *all* available antigens; if the ratio of these quantities is less than some predefined percentage d (the deletion threshold), then the antibody is deleted. However, as in the COSDM model, an antibody is allowed to exist for at least n epochs after creation in order to give it an opportunity to survive. Furthermore, a caveat is applied that if an antibody uniquely recognises at least one antigen, then it is allowed to remain.

6.7.3.3 Results

A series of experiments was performed in which SODSM was used to try and cluster the half-schema, quarter-schema and eighth-schema data used throughout this thesis. Each dataset contained 200 antigens, and in each experiment SOSDM was initialised with 2 antibodies. The stagnation threshold s is set to 10 iterations, and the deletion

Data	No. Antibodies (original data)	Average No. Antibodies using SOSDM	Average Recall Accuracy
half-schema	2	2.29	49.41
quarter-schema	4	6.75	44.77
eighth-schema	8	10.06	42.40

Table 6.7: The table shows the average number of antibodies required and corresponding accuracy of recall for clustering data-sets with a dynamic SOSDM algorithm

threshold d was varied as described below. At the end of each experiment, the best recall accuracy and the corresponding number of antibodies in the system are recorded. Each experiment was run 100 times and the results averaged. Initial experiments using the half-schema data showed that the actual value of the deletion threshold parameter d was unimportant in terms of the recall accuracy the system achieved and the average number of antibodies used, however it had a large effect on the number of times antibodies were deleted from the system and then subsequently re-added, hence a careful choice is necessary in order to make the system efficient. These results are shown in table B.9, appendix B, which clearly indicates that for this data, a large increase in the instability of the system occurs when the deletion threshold rises above 0.3. However, for all values of d, the system always produces its best results when the number of antibodies is on average 2, as desired. Experiments with the quarter-schema data and eighth-schema data were performed with d set to 0.25. The average number of antibodies required to give the best recall is shown in table 6.7.

The number of clusters in each case is sensible — although the original data-sets were created using 2,4 and 8 schemas and hence nominally contain the corresponding number of clusters, these clusters are somewhat arbitrary. Recall that the data is created by randomly filling in wild-cards in a set of schemas, therefore the formation of other clusters is likely, especially when the defined length of the schemas is short. Thus, with the half-schema data, the data is most accurately recalled using 2 or 3 clusters, closely matching the original schemas, whereas in the eighth-schema data, more accurate recall is gained by using more than the 8 clusters that the data was generated

from.

6.8 Conclusions

This chapter has presented a new model for clustering data, combining ideas from immunology, self-organising maps, and associative memories. A thorough investigation of the model's performance has been carried out, in particular:

- The ease of calibrating the model, i.e. its sensitivity to internal parameters
- The performance of the model on a set of benchmark data
- The limitations of the model with respect to characteristics of the data-sets on which it operates
- The performance of the model in non-stationary environments

The results have shown that SOSDM outperforms both CE-POTTER and COSDM on the benchmark data, and that it is straightforward to calibrate. Furthermore, it is fast and reliable — the benchmark datasets were clustered accurately in fewer than 100 iterations of the controlling algorithm. The performance of the system on data-sets other than the benchmark data was also promising, in that the system appears to scale well with both the size of the data-sets and the length of the data items within the data-sets. The problem of clustering data-sets in which there is an uneven distribution of data within clusters was also addressed. The results of this investigation showed that recall of very small clusters could be improved by extending the model to include Fitness Proportionate Data Selection, FPS, however, this was at the expense of decreasing the accuracy of recall of the entire data-set. Encouragingly though, SOSDM still performs better than both CE-POTTER and COSDM on these data-sets. Clearly this is something that needs to be addressed more fully in the future, as real data-sets are unlikely to contain data evenly distributed between clusters. A possible alternative to FPS would be to utilise Dynamic Subset Selection of data-items, a method proposed by Gathercole in [Gathercole and Ross, 1994]. In this paper, Gathercole *et. al* suggest

a method for selecting training examples during evolution of a classification function-tree using Genetic Programming. Cases are selected on the premise that it is of benefit to focus attention on those cases which are currently *difficult*, i.e. are currently mis-classified, and also on those cases which have not been looked at for several iterations. Thus cases are selected for training with a bias that is based on both the difficulty and 'age' of the case. However, both this method and FPS suffer from the drawback that they effectively implement a type of supervised learning, whereas ideally an immune system based model would be truly unsupervised. Furthermore, they both require the recall accuracy of each item to be calculated following every iteration — this is not always desirable and adds a considerable time overhead, especially if the data-set is large.

Experiments performed in dynamic environments showed that SOSDM rapidly adapts to data that is changing within *fixed* clusters, being able to return to its previous levels of accuracy within a few iterations. It responds less well to the appearance of entirely new clusters, though as previously noted, this is an unlikely scenario in a real-world situation. Certainly new clusters will appear over the course of time, however it is likely to be a slow and gradual process. Furthermore, the quality of the response is somewhat dependent on the characteristics of the dataset. One radical solution would be to periodically restart the algorithm from scratch using the current data; this is feasible given the short time-scales required to run the algorithm, however it has the major disadvantage that all historical information contained in the counters is lost. New methods of dealing with the formation of new clusters will be tackled in the future, and are likely to be closely related to the mechanisms incorporated in the algorithm for adding and deleting antibodies dynamically.

Currently the system incorporates simple mechanisms for determining when to delete and add antibodies. The addition mechanism is fairly crude in that it detects stagnation of the system simply by monitoring movement of the antibodies, and then adds a new antibody in a random position. This could be made more sophisticated by adding the new antibody in a part of the input space which is not well represented by the current system, for example by interpolating between the antibodies with the largest accumulated error, as in Fritzke's GNG algorithm. The criterion for determin-

ing whether an antibody should be added could also be improved. A simple suggestion would be to add a new antibody if it is determined that the distance between some antigen and the closest antibody is greater than some constant d. However, an alternative proposal which it is intended to follow up in the future would be to cause antibodies which only exhibit weak binding (due to data appreaing far away) to emit a *distress signal*. On the other hand, antibodies binding strongly to antigens could emit a *contentment signal*. Monitoring the overall level of distress in the system could then prompt creation of a new antibody within the system. This has a direct analogy with the danger model proposed by [Matzinger, 1994b] in which it is claimed that cells emit a danger signal when faced with invading and dangerous pathogens. Furthermore, this system is not particularly information-intensive and hence is appealing from the computational perspective.

In summary, SOSDM appears promising as a model for clustering data, both stationary and non-stationary. It has addressed the problems inherent in COSDM of dealing with fixed, pre-determined radii and of potentially failing to recognise proportions of the data, and performs very well on the benchmark datasets. It has also been shown to be scalable. Though improvements need to be made to the mechanisms which allow it to operate in a non-stationary environment, the experiments have shown that in its current state, it is capable of clustering moving data-sets with some success, even in extreme conditions. Finally, the model itself has moved closer to embodying the basic principles of the immune system, in that it is self-organising and unsupervised.

Chapter 7

Conclusion

7.1 Overview

This chapter summarises the work presented in the preceding six chapters. It addresses the question of to what extent the original aims of the thesis have been met, and discusses the usefulness of the immunological metaphor in the context of other similar systems. Finally, some suggestions for future work are presented.

7.2 Were the aims achieved ?

The aim of this thesis as stated in the introductory chapter was to assess whether the immune metaphor provided features which distinguished it from other biological metaphors, and to attempt to categorise the types of problem area where application of the metaphor might prove advantageous. In order to do this, two application areas were examined, those of scheduling and data-clustering (both stationary and non-stationary). For both problem domains, two different models were developed, incorporating a number of immunological principles. The success of each model in its relevant problem domain has been discussed in detail at the end the chapter in which each model was introduced. In this section, some general conclusions are drawn regarding the design of each model.

Table 7.1 lists the distinguishing properties of the immune system, as described by

	Artificial Immune System			
	SCHED1-IS	SCHED2-IS	COSDM	SOSDM
Recognition	yes	yes	yes	yes
Feature-Extraction	yes	yes	yes	yes
Diversity	yes	yes	no	yes
Learning	yes	yes	yes	yes
Memory	yes	yes	yes	yes
Distributed detection	no	no	possible	possible
Self-regulation	no	no	no	yes
Threshold-mechanism	no	yes	yes	yes
Co-stimulation	no	no	no	no
Dynamic protection	no	no	yes	yes
Probabilistic detection	yes	yes	yes	yes

Table 7.1: The table identifies which of the distinguishing features of the biological immune system are present in each of the models developed in this thesis

[Dasgupta, 1998] and presented in chapter 1. For each of the four new AIS models developed in this thesis, the table identifies which of the distinguishing features of the biological immune system are present in each model. Each model is now discussed in turn in more detail with reference to this table.

7.2.1 Scheduling Models, SCHED1-IS and SCHED2-IS

The $SCHED1-IS$ and $SCHED2-I$ algorithms both involve two distinct phases in schedule production. In the first phase, schedule building blocks are derived (via an evolving immune library in the case of $SCHED1-IS$ and by finding blocks that match existing schedules in the case of $SCHED2-IS$). In the second phase, the building blocks are combined to produce new schedules (randomly for $SCHED1-IS$ and using mechanisms based on immunological principles in the case of $SCHED2-IS$).

In $SCHED1-IS$, there is no direct analogy of the usual *recognition* process that occurs between an antibody and an antigen — an indirect recognition process occurs

however when the schedule builder produced as a result of combining segments from the evolved libraries is used to form a schedule based on the conditions defined by the antigen. The quality of the actual schedule produced as a result of binding the schedule-builder with the antigen defines the strength of the recognition process. During the evolution of the schedule fragments in $SCHED2-IS$, the recognition process is explicit; schedule fragments are directly matched against existing schedules in order to generate the building blocks from which new schedules can later be derived.

Both models exhibit *feature detection*, the features detected being useful fragments of schedule, or schedule builder. Both models also exhibit *diversity*, in that a small number of schedule fragments can result in a large number of schedules, owing to the combinatoric manner in which fragments are used to produce entire schedules. *Learning* is also apparent in both models — in each case, a genetic algorithm is used to learn which schedule fragments are required to make up the immune system. The models thus also utilise *memory*, which is simply the store of schedule building blocks.

Neither $SCHED1-IS$ or $SCHED2-IS$ provide *distributed detection* in the same manner as the biological immune system which must provide physically distributed detection in order to detect invaders entering any part of the body. However, it could be argued that they are robust to individual points of failure within each system; as each system consists of multiple schedule fragments, loss of individual fragments does not render either system useless. Therefore, they exhibit the properties that distributed detection confers. Similarly, neither $SCHED1-IS$ nor $SCHED2-IS$ are *self-regulating*. Use of a genetic algorithm with an associated fitness function is clearly an external regulatory mechanism and hence the systems cannot be considered to be in any sense self-regulating.

$SCHED1-IS$ does not contain any kind of *threshold mechanism* — a schedule can always be produced by combining an antibody, i.e. a schedule-builder, with an antigen (the scheduling conditions) and although the quality of resulting schedule feeds back into the fitness function of the genetic algorithm, it does not prevent 'binding', (i.e. formation of a schedule) occurring. On the other hand, an explicit threshold mechanism is incorporated in $SCHED2-IS$, below which a match does not occur.

Co-stimulation, i.e. the presence of a 'second signal' reinforcing the match between

antigen and antibody is not utilised in either $SCHED1-IS$ or $SCHED2-IS$. As both systems are essentially static, i.e. once a set of schedule fragments has been produced it remains constant, there is little need for such a mechanism. However, such a mechanism might prove useful if the models were adapted such that the set of schedule fragments was continuously adapted, so that fragments which do not prove useful in generating good schedules were gradually replaced by new fragments. In this case, co-stimulation would be provided to indicate that a schedule fragment contributed to producing a good schedule and thus prolong its 'life' in the set of all fragments. The fact that both scheduling models are static systems also means that there is no feature in these models that has a direct analogy with the *dynamic protection* feature observed in the natural immune system, which increases the cover provided by the immune system over time.

Probabilistic Detection in the natural immune system implies that a lymphocyte can bind with several kinds of structurally related antigen. Due to the absence of a threshold-mechanism for $SCHED1-IS$, any antibody can bind with any antigen, though the resulting objective function for the schedule may be low. For $SCHED2-IS$, probabilistic detection is implicit; a schedule fragment can match a set of antigens owing to the presence of wild-cards within each fragment, and does not match other antigens at all due to the use of a matching threshold.

7.2.2 Data-Clustering Models, COSDM and SOSDM

COSDM and SOSDM are both based on the concept of the Sparse Distributed Memory, but are hybridised with other methodologies in order to provide an adaptive system. Thus, COSDM is a hybrid of an SDM and a genetic algorithm, whereas SOSDM is a hybrid of an SDM and a self-organising map. In chapter 1 of this thesis, it was stated that one of the questions that this thesis hoped to address was *"to what end does the immunological metaphor provide an analogy that cannot be provided by another less seductive labelling"* therefore this section addresses the point as to whether it is justifiable to refer to the hybridised systems presented as 'immune systems'.

Any basic introduction to biological immunology almost always begins with a discussion of the respective roles of antibodies and antigens, therefore clearly these

features must be able to be identified in COSDM and SOSDM as a first step to calling them immune systems. The two preceding chapters have shown that for a data-clustering system these roles *can* be identified. In both COSDM and SOSDM, data can be considered as antigen, and also in both systems, the 'centres' defining the SDM play the role of antibodies, therefore the systems contain the essential core components. Therefore, it seems reasonable to argue that *if* these components are there, *and* that the systems exhibit the defining features of the immune system as given by [Dasgupta, 1998] then it is justifiable in referring to COSDM and SOSDM as immune systems. [Smith et al., 1999] has already shown that an SDM is a member of the same class of associative memories as immune memory, however, this fact alone is not sufficient to justify calling an SDM-based system an immune system — a comparison of table 7.1 and table 2.1 in chapter 2 shows that there are features exhibited by the immune system that are not observed in Smith's mapping between the SDM and the immune system. The relationship between each of the features in table 7.1 and COSDM/SOSDM is now discussed.

Both the COSDM and SOSDM models clearly incorporate a *recognition* mechanism — antibodies bind to antigen only when the affinity between the two is considered to be sufficiently high. In the case of COSDM, this involves the use of a *threshold mechanism*, i.e. the recognition radius, which directly determines whether the antibody recognises the antigen or not. In SOSDM, the concept of the recognition radius is replaced by the affinity-threshold, which effectively acts as a threshold mechanism in that it determines whether any binding at all can take place. The difference between the models is that in SOSDM, antigens are always distributed to at least one antibody, whereas in COSDM, the threshold mechanism can prevent recognition of an antigen by any of the antibodies. In both cases, recognition is clearly *probabilistic* — an exact match between an antibody and an input data item (antigen) is not required in order for recognition to occur.

Both COSDM and SOSDM perform *feature extraction* — by performing clustering of the data, the system is indirectly detecting features in the data. In each system, an antibody describes the feature present in the cluster its represent by virtue of its description (i.e. address) and its associated set of counters.

Diversity as described in [Dasgupta, 1998] is the usage of a combinatoric process in order to generate a diverse set of lymphocytes that can bind to any antigen, whether known or unknown. COSDM does not exhibit true diversity, as discussed in chapter 5 — if suitable recognition radii cannot be found, then it is possible for a COSDM system to fail to recognise some antigens. SOSDM on the other hand does not utilise a combinatoric mechanism to confer diversity, but achieves the same result via a competition between antibodies for recognising an antigen, which always results in the antigen being recognised by at least one antibody, therefore an SOSDM model will always recognise both known and unknown antigen.

Both clustering algorithms are capable of *learning*, in order to find the best antibodies to accurately describe the clusters present in the data. In COSDM this is achieved via a genetic algorithm, in SOSDM by using a similar learning algorithm to that found in self-organising maps. By definition, the models exhibit *memory* — they are examples of associative, content-addressable memories. Furthermore, they both have the potential for offering *distributed detection*. Antibodies can be physically distributed across servers, and the systems are robust to loss of one or more of the antibodies without adversely affecting performance, as in each case, antigens can be bound by more than one type of antibody.

Co-stimulation is not modelled by COSDM or SOSDM. In principle, it could be included in both models to reinforce addition of new antibodies when the system becomes static. Both models however exhibit *dynamic protection*, in that the systems can respond to dynamic changes in the environment. This is due to the ability of the antibodies to 'move', i.e. adapt their definition, in response to changes in the environment.

Both systems exhibit probabilistic detection. The dynamic manner in which antibodies are added to the system in both COSDM and SOSDM result in new antibodies being added at random positions in the systems, and thus confers an element of probabilistic detection.

Finally and most importantly, SOSDM embodies one of the most important principles of the biological immune system listed in table 7.1, that it is *self-regulatory*. This feature is not displayed by COSDM which by use of a genetic algorithm to control its evolution requires an external fitness function.

In addition, COSDM and SOSDM also incorporate some other biological features not mentioned in table 7.1. Binding of antigens to specific antibodies based on the attraction of the antibody for the antigen is consistent with the idea of *shape-space*, first introduced as an abstract concept by [Perelson and Oster, 1979]. In this model, antibodies and antigens are considered as points in a 'shape-space', and antibodies within the affinity cut-off for clonal selection by an antigen form a ball in the shape-space known as the ball of stimulation. [Perelson and Oster, 1979] attempted to make the shape-space quantitative by representing antibodies and antigen with real-valued coordinates, however an alternative to this kind of Euclidean shape-space is the Hamming shape-space used by for example [Farmer et al., 1986, Hightower et al., 1995, Perelson et al., 1996]. Thus, in COSDM, the size of the shape-space is directly determined by the recognition radius of each location, and the corresponding quantity can be calculated in SOSDM by determining the maximum distance between an antibody and an antigen recognised by it. Furthermore, due to its self-organising nature, SOSDM also exhibits the meta-dynamic behaviour observed in immune-networks and discussed in chapter 2. Every time the antigen data is presented to the system, the definition of the antibodies may be perturbed, but the system eventually settles into a stable representation of the current input data, representing the core clusters.

Thus, of the four models presented in this thesis, SOSDM comes closest to modelling all of the features of the real immune system. Everyone of the features listed in table 7.1 is apparent in the model, (or could potentially be added) and it contains the core components of an immune system, i.e. antigens and antibodies, and the ability of one species to recognise the other. Therefore, it seems justified to label the system as an immune system. Furthermore, the next section shows the hybrid SOSDM/COSDM systems encapsulates precisely the properties that are required of a data-clustering system, and can offer some advantages over standard clustering algorithms, and that therefore approaching the design of the system from an immunological perspective has proved beneficial.

7.3 Is Immunology a Useful Metaphor ?

In the 14th century, William of Occam, a logician and Franciscan friar is said to have remarked *"Entia non sunt multiplicanda praeter necessitatem"*, or "Entities should not be multiplied more than necessary". This guiding principle, known as Occam's Razor, is usually interpreted as meaning 'the most simple explanation is the one to be preferred'. Thus, with this in mind, it is necessary to consider whether, in the context of the work presented in this thesis, the immune system has proved a useful metaphor, or whether the same problems could have been tackled by other methods. In this section, we examine some of the systems to which immune systems have been compared, to see if they could have been applied to the problems considered.

7.3.1 Other approaches to scheduling

In [Farmer et al., 1986], the analogy between the immune system and *classifier systems* was discussed in detail. The idea of the classifier system was first introduced by Holland in [Holland et al., 1986], and there exist many variations of the basic system. However, in its simplest form, it consists of a number of *classifiers*, which are simply rules comprising of a *condition* and an *action*. If the condition part of any rule is matched by a message from the environment, then the rule bids to be able to fire and hence execute its action. Each bid is a fraction of the rule's associate strength — the winning rule gets to fire and executes its action, and then may earn some reward from the external environment. This reward is paid to the winning rule, though in some more elaborate systems, the winning rules also pay some of their strength to those rules which made it possible for them to fire, in a kind of 'trickle-down information economy'. New classifiers are generated either at random, or using a GA, which periodically runs in order to generate new rules and replace existing ones of low strength. The paper by Farmer illustrates the analogy in detail. Could the type of scheduling problem described in chapter 3 be tackled by a classifier system approach, rather than using the immune metaphor ?

First note that the analogy as described by Farmer applies to *immune-networks*, and not necessarily to the type of immune model presented in chapter 3. In this chap-

ter, scheduling problems were solved by recombining evolved 'segments' of schedules, with the rationale that the schedule segments (whether stored in libraries as in $SCHED1-IS$ or in a population as in $SCHED2-IS$) captured prior useful experiences, and therefore reduced the size of the potential search space. When formulating the problem in this way, it is difficult to see how the problem could be mapped onto a classifier system approach. However, there are potential approaches to re-scheduling that could benefit from use of a classifier system. For example, if the 'condition' part of the classifier was used to match the current state of the scheduling environment, i.e. the jobs yet to be scheduled, information pertaining to arrival-dates and due-dates, and current machine usage, then the associated action of the rule could specify a heuristic to be applied in order to select the next job to be scheduled. This kind of approach has already been adopted in work in which the author has been involved, [Ross et al., 2002] for solving bin-packing problems, and could easily be extended to scheduling. Clearly, an immune-system analogy is also relevant here, the important functions exhibited by the classifier in this example are its ability to *match* (imprecisely) information in the environment, to *learn* suitable associated heuristics, and also to *adapt* as the environment changes. These functions are of course exhibited by the immune system — though it would be straightforward to use the metaphor to find 'rules' that matched the environment, some further thought would have to be applied in order to find a suitable method for associating the matching rules with the correct heuristic.

Another approach which immune-systems have been compared to is that of *case-based-reasoning systems* (CBR), first discussed by [Hunt et al., 1995]. A CBR system relies on having a representative database of cases which is efficiently organised and has suitable mechanisms for retrieving the cases. The manner in which memory is organised and the way in which cases are retrieved is crucial, and yet can be very application dependent. It is often difficult to identify the most suitable organisation prior to actually developing and experimenting with the system, which results in long development times and can severely restrict the usefulness of the approach. However, [Hunt et al., 1995] argue that the immune system exhibits exactly the properties required of a CBR system: the immune system is inherently case-based, it relies on a content addressable memory, it contains a general pattern matching mechanism, and

most importantly, the memory is self-organising. In addition, the immune system possesses the ability to 'forget' cases which are no longer useful, therefore improving the efficiency of the system.

[Hunt et al., 1995] compared an immune-network to several other forms of case-memory, namely linear memories, hierarchical memories, nested memories, decision-trees, and knowledge-guided indexing. The analysis showed when compared to these memories, *only* the immune system has a structured memory, is inherently incrementally adaptable, can automatically create its memory structure without a memory 'designer' identifying the appropriate structure, is inherently self-organising and provides an implicit mechanism for case-forgetting. Furthermore, when considering retrieval mechanisms, the immune system can focus search towards similar cases and can handle noisy or missing data. None of the previously mentioned forms of memory exhibit both these properties. However, the immune system has one potential drawback in that it is *not* deterministic, and does not necessarily return the same result given the same inputs. This is in contrast to the other forms of memory considered.

Is then, a CBR system a suitable methodology for tackling scheduling problems ? Certainly, in the problem described in chapter 3, a database of previous cases, i.e. schedules, could be built up. Straightforward matching algorithms could be utilised to match partial schedules and environmental conditions to cases in the databases, as in general we would only be dealing with integer representations. Thus, it is conceivable that a CBR approach might be adopted in some circumstances. However, a CBR approach has a major drawback in that it is not possible to produce entirely *new* schedules from such a system, that do not resemble existing cases in the database. This is not true of an immune system approach in which the building blocks from which schedules are built can be recombined in many ways to produce novel schedules which are appropriate for the current conditions. Thus, this fact might prove a significant advantage for an immune-system rather than CBR approach in this case.

7.3.2 Other Approaches to Data-Clustering

In [Timmis et al., 1999], the author compares the performance of an immune-network algorithm for clustering with a simple clustering technique known as *Single Linkage*

Clustering and a Kohonen Network on Fisher's Iris data set. He concludes that "*the AIS provides a more diverse representation of the data than the other two methods which is useful for classifying unseen data and variations on unseen data, and that in addition, the AIS is effective at allowing the user to explore the evolved network to gain a fuller understanding of the makeup of the dataset, thus aiding the data-mining and analysis process.*". Therefore, it is necessary to consider whether either a self-organising map approach, as typified by the Kohonen network, could have been used to cluster the artificial data-sets described in chapter 4 and tested with the COSDM and SOSDM models.

The Kohonen network can clearly be applied to cluster static data sets, although [Timmis et al., 1999] found that it was unable to discover the three distinct classes present in the Fisher data set, whereas his AIS algorithm was able to correctly identify three separate clusters. However, due to the arguments outlined in chapter 2, section 2.3, it is unsurprising that the Kohonen algorithm could not correctly cluster this data — an unsupervised clustering algorithm could not correctly classify this data due to the geometric intermingling of the data classes. However, despite this, it is clear that Timmis's *AINE* algorithm *does* visually produce three distinct clusters whereas the Kohonen algorithm does not, therefore it is of benefit in discovering general features in data-sets, even if the classification accuracy is not 100%.

The Kohonen algorithm in its original form is not suitable for clustering non-stationary data, although extensions to it have been suggested, for example [Abrantes and Marques, 1998] give a framework for clustering dynamic image and video data based on the Kohonen algorithm. Other algorithms for clustering dynamic data also exist; Fritzke's Growing Neural Gas algorithm was described in chapter 6, and as mentioned, has been applied to cluster slowly changing distributions, [Fritzke, 1997a]. It would be interesting to examine how this algorithm performs on the data used in this thesis — this work will be performed in the future. A point to note however is that clustering algorithms such as GNG and Kohonen rely on calculating the Euclidean distance between vectors in order to cluster them — the data-sets described in chapter 4 consist of binary data, and hence it might be more appropriate to modify these algorithms to use Hamming distance as a measure of similarity between

vectors.

However, the SOSDM based on a combination of ideas from the immune-system and Sparse Distributed Memories may offer some advantages over the standard clustering algorithms. The most obvious advantage is that SOSDM is an *associative memory*, as well as a means of clustering data. Thus, data can be stored and retrieved from the memory if necessary, and new data easily be categorised on presentation to the memory. An important feature of the SOSDM not provided by the other clustering techniques is that the combination of the address and counters defining the centre of each hard location in the SOSDM provides a concise description of the cluster itself; this information is not available from either the GNG or Kohonen algorithms. The method also provides a mechanism for identifying anomalous data, either by monitoring the system for appearance of new hard locations and hence clusters, or tracking large movements in existing clusters. A further point to consider is also that the SOSDM algorithm allows data to belong to more than one cluster — in real databases, clusters will rarely be able to be completely isolated, therefore this feature may be useful.

Finally, [Dasgupta, 1997] has made clear the relationship between immune systems and neural networks. A neural network is not a suitable tool for clustering data, though could be used to perform anomaly detection in data by training the network with examples of items from known clusters. This obviously requires previous classification of the data into clusters. However, the idea is mentioned here as a *recurrent* network could provide a means of performing anomaly detection in a non-stationary environment with continuously adapting weights.

In summary, the models presented in this thesis do suggest that the immune metaphor might provide features not apparent in other more established techniques for tackling both scheduling problems and dynamic data-clustering problems. It seems clear that the way forward is not to attempt to adhere strictly to biological principles when building artificial models, but to use them as inspiration for suitable computational techniques, modifying principles and adding new ones where necessary. This is not a drawback — many successful neural network applications are built on the back of the back-propagation algorithm, yet it is clear that this algorithm is not utilised by

nature in biological neural systems. Thus, immunology can provide a useful addition to the computer-scientists armoury of metaphors for solving complex problems, and the field is deserving of further attention.

7.4 Suggestions for Future Work

It would seem most fruitful to extend the SOSDM model, as noted above, it embodies most of the principles of the biological immune system and the experimentation performed so far with it indicates its potential value. An obvious extension would be to add a visualisation mechanism to the system, so that the topology of the memory could be observed, for example as in the work by [Timmis, 2000b]; whilst a two dimensional map would have no meaning in terms of the x-y coordinates of points on the map, it would be useful to observe the relationships between clusters and between points within the clusters. Along similar lines, it would be useful for the system to be able to report information concerning movement of the centres over time, which would indicate movement of trends in the data more explicitly. If the system is to be used to perform anomaly detection or to classify new items of data, then further reporting mechanisms must be added; thus, warnings could be produced when either a new centre is added or an existing centre moves significantly when performing anomaly detection, or the centre recognising a data-item should be reported when classifying new data-items.

Another area in which the algorithm could be improved is the manner by which new centres are added and deleted. This would make the system more useful when clustering data sets in which no sensible guesses can be made about the number of clusters likely to be present, and would also improve the pattern-tracking capabilities of the system. A possible way that this could be implemented would be to introduce a 'detector lifetime' as in [Hofmeyr and Forrest, 2000] which would result in centres not recognising new data over a long period of time being removed.

Finally, in order for the SOSDM to ultimately prove useful in clustering real-world data, a method must be formulated for representing non-binary data. The original method in which counters are updated and are used to retrieve items from the memory

is suitable only for binary representations, therefore, a mechanism must be found for modifying this algorithm, or alternatively for mapping real-valued data into binary form so that the original algorithm can be used.

Appendix A

Coincidences in permutations and schedules

Thanks to [Ross, 2002] for providing this proof.

Given two permutations of the numbers $1...n$, what is the expected number of coincidences between them. A coincidence is when the numbers in position i are the same. Without loss of generality, you can suppose that the first permutation is $1..n$ itself, so the question is then as follows: given a random permutation, what is the expected number of instances in which position i contains i ?

Let the random variable $f(P)$ be the number of coincidences in permutation P. If $f_k(P)$ is the random variable which is 0 or 1 according to whether position k contains k then

$$f(P) = \sum_{k=1}^{n} f_k(P)$$

because the $f_k(P)$ are independent (the fact that position k contains k cannot affect whether other positions i contain i). So, in expectation,

$$E(f(P)) = \sum_{k=1}^{n} E(f_k(P))$$

but $E(f_k(P)) = 1/n$ because $(n-1)!$ permutations out of $n!$ contain k in position k. Thus

$$E(f(P)) = 1$$

Thus, no matter how long the permutation, you expect just one item to be in its place!

What is the variance of f? Consider $E(f^2)$):

$$\begin{aligned} E(f^2) &= \sum_{j=1}^{n}\sum_{k=1}^{n} E(f_j)E(f_k) \\ &= \sum_{j=1}^{n}\sum_{k=1} E(f_j f_k) \\ &= \sum_{j=1}^{n} E(f_j^2) + 2\sum_{1\le j\le k\le n} E(f_j f_k) \end{aligned}$$

But f_j is just 0 or 1, so $f_j^2 = f_j$, so $E(f_j^2) = E(f_j) = 1/n$. Also, $E(f_j f_k)$ is the probability that a permutation has both j and k in place, namely, $(n-2)!/n! = 1/n(n-1)$. Therefore,

$$E(f^2) = 1 + 2 \times (n(n-1)/2) \times 1/n(n-1) = 2$$

and the variance of f is therefore $E(f^2) - E(f)^2 = 2 - 1 - 1$.

Appendix B

Experimental results obtained using SOSDM

Antigens	Probability that true means differ	CE-Potter	SO-SDM
50	0.999773	49.446000	50.142000
100	0.999836	48.872000	49.496000
150	0.999999	48.828667	49.417333
200	0.999883	48.710000	49.328500
250	1.000000	48.860000	49.220400
300	0.999999	48.577333	49.147666
350	0.999993	48.520000	49.062286
400	0.999996	48.512250	49.081250
450	0.995579	48.334889	48.929778
500	0.999992	48.434400	48.983600

Table B.1: T-tests comparing CE-Potter and SOSDM for Half-Schema Experiments

Antigens	Probability that true means differ	CE-Potter	SO-SDM
50	0.999998	43.872000	45.016000
100	1.000000	43.073000	44.201000
150	1.000000	42.568667	43.528666
200	1.000000	42.401500	43.567000
250	1.000000	42.528400	43.402800
300	1.000000	42.301333	43.251333
350	0.999991	42.394000	43.134286
400	1.000000	42.458000	43.143500
450	1.000000	42.273555	43.142889
500	1.000000	42.273555	43.142889

Table B.2: T-tests comparing CE-Potter and SOSDM for Quarter-Schema Experiments

Antigens	Probability that true means differ	CE-Potter	SO-SDM
50	0.999863	43.966000	44.826000
100	1.000000	42.246000	43.150000
150	1.000000	41.776000	42.619333
200	1.000000	41.439500	42.228000
250	1.000000	41.058400	41.728400
300	1.000000	40.868333	41.494000
350	0.999999	40.762571	41.254286
400	0.999999	40.578000	41.053250
450	0.999991	40.586222	40.895333
500	0.999997	40.465600	40.821400

Table B.3: T-tests comparing CE-Potter and SOSDM for Eighth-Schema Experiments

subset size for FPS	p means differ	Data stored in SOSDM at each epoch	
		Standard SOSDM	FPS
10	1.000000	50.275636	49.820636
20	1.000000	50.275636	49.868091
30	1.000000	50.275636	49.919455
40	1.000000	50.275636	49.978864
50	1.000000	50.275636	49.965364
60	1.000000	50.275636	49.983727
70	1.000000	50.275636	50.000455
80	1.000000	50.275636	50.039227
90	1.000000	50.275636	50.017182
100	1.000000	50.275636	50.038636
110	1.000000	50.275636	50.044545
120	1.000000	50.275636	50.023000
130	1.000000	50.275636	50.023636
140	1.000000	50.275636	50.071591
150	1.000000	50.275636	50.068000
160	1.000000	50.275636	50.063773
170	1.000000	50.275636	50.078273
180	1.000000	50.275636	50.089773
190	1.000000	50.275636	50.081455
200	1.000000	50.275636	50.087364

Table B.4: Table shows the probability that the mean fitness *across entire dataset* obtained using FPS is statistically different than the mean fitness obtained when *all* data is stored at each epoch

subset size for FPS	p means differ	Data stored in SOSDM at each epoch	
		All data	FPS
10	1.000000	44.238182	46.193636
20	0.999552	44.238182	45.450909
30	0.999974	44.238182	45.674545
40	0.999999	44.238182	45.889091
50	1.000000	44.238182	45.926364
60	0.999985	44.238182	45.686364
70	1.000000	44.238182	46.004545
80	0.999987	44.238182	45.639091
90	0.999944	44.238182	45.539091
100	0.998712	44.238182	45.297273
110	0.999276	44.238182	45.340909
120	0.999964	44.238182	45.570000
130	0.999976	44.238182	45.653636
140	0.999778	44.238182	45.458182
150	0.999979	44.238182	45.624545
160	0.999979	44.238182	45.586364
170	0.999998	44.238182	45.689091
180	0.999867	44.238182	45.498182
190	0.999984	44.238182	45.605455
200	0.999441	44.238182	45.333636

Table B.5: Table shows the probability that the mean fitness *of smallest cluster* using FPS is statistically different than the mean fitness obtained when *all* data is stored at each epoch

Number of defined bits	Index of update	Average Lag	Minimum Lag	Maximum Lag
8	2	159.200000	100	216
	3	276.600000	187	364
	4	407.000000	280	497
	5	611.000000	372	846
16	2	100.000000	100	100
	3	138.200000	100	231
	4	128.200000	100	165
	5	268.600000	143	348
32	2	146.800000	100	206
	3	324.000000	201	504
	4	121.600000	100	208
	5	615.250000	459	819

Table B.6: Data sets containing 2 clusters: Average lag between updates

Number of defined bits	Index of update	Average Lag	Minimum Lag	Maximum Lag
8	2	196.400000	143	243
	3	250.600000	196	321
	4	333.800000	210	372
	5	389.600000	238	567
16	2	143.400000	115	180
	3	154.600000	119	211
	4	180.800000	100	259
	5	255.000000	169	317
32	2	304.000000	214	394
	3	264.500000	125	404
	4	144.000000	143	145
	5	100.000000	100	100

Table B.7: Data sets containing 5 clusters: Average lag between updates

Number of defined bits	Index of update	Average Lag	Minimum Lag	Maximum Lag
8	2	183.600000	137	283
	3	303.200000	236	476
	4	468.200000	307	678
	5	509.750000	354	635
16	2	228.000000	191	255
	3	475.200000	379	597
	4	807.500000	586	979
32	2	178.000000	119	218
	3	158.000000	100	213
	4	211.200000	162	257
	5	150.400000	121	175

Table B.8: Data sets containing 5 clusters: Average lag between updates

Deletion Threshold	Average No. Centres	Average Recall Accuracy	Average No. Centres Deleted	Average No. Centres added
0.1	2.41	49.46	1.84	2
0.2	2.25	49.43	1.92	2
0.3	2.61	49.72	7.68	2
0.4	2.52	49.70	7.73	8
0.5	2.54	49.69	7.73	8
0.6	2.62	49.83	7.61	8
0.7	2.67	49.74	7.67	8
0.8	2.50	49.70	7.72	8
0.9	2.65	49.85	7.58	8
1.0	2.59	49.70	7.72	8

Table B.9: Variation in recall accuracy and in the number of centres added and deleted with deletion threshold

Bibliography

[Abrantes and Marques, 1998] Abrantes, A. and Marques, J. (1998). A method for dynamic clustering of data. In *British Machine Vision Conference*.

[Baldwin, 1896] Baldwin, J. (1896). A new factor in evolution. *American Naturalist*, 30:441–451.

[Beasley, 1990] Beasley, J. (1990). Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

[Bierwirth et al., 1995] Bierwirth, C., Kopfer, H., Mattfeld, D. C., and Rixen, I. (1995). Genetic algorithm based scheduling in a dynamic manufacturing environment. In *Proc. of 1995 IEEE Conf. on Evolutionary Computation*, pages 439–443, Piscataway, NJ. IEEE Press.

[Booker, 1985] Booker, L. (1985). Improving the performance of genetic algorithms in classifier systems. In Grefenstette, J., editor, *Processings of the an International Conference on Genetic Algorithms and their Applications*, pages 80–92.

[Burnet, 1959] Burnet, F. (1959). *The clonal selection theory of immunity*. Vanderbilt University Press, Nashville, TN.

[Carter, 2000] Carter, J. (2000). The immune system as a model for pattern recognition and classification. *Journal of the American Medical Informatics Association*, 7(3):28–41.

[Celada and Seiden, 1992] Celada, F. and Seiden, P. (1992). A computer model of cellular interactions in the immune system. *Immunology Today*, 13(2):56–62.

[Cooke and Hunt, 1995] Cooke, D. E. and Hunt, J. (1995). Recognising promoter sequences using an artificial immune system. In *Proceedings of Intelligent Systems in Molecular Biology*, pages 89–97, CA. AAAI Press.

[Costa et al., 2002] Costa, A., Vargas, P., Von Zuben, F., and Franca, P. (2002). Makespan minimisation on parallel processors: an immune based approach. In *Proceedings of the Special Session on Artificial Immune Systems in the 2002 Congress on Evolutionary Computing*.

[Dasgupta, 1996] Dasgupta, D. (1996). Using immunological principles in anomaly detection. In *Proceedings of Artificial Neural Networks in Engineering, (ANNIE'96)*, St Louis, USA.

[Dasgupta, 1997] Dasgupta, D. (1997). Artificial neural networks and artificial immune systems: Similarities and differences. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*.

[Dasgupta, 1998] Dasgupta, D. (1998). *Artificial Immune Systems and Their Applications*, chapter An Overview of Artificial Immune Systems and Their Applications, pages 3–18. Springer-Verlag.

[Dasgupta and Forrest, 1999] Dasgupta, D. and Forrest, H. (1999). Artificial immune systems in industrial applications. In *Proceedings of the International conference on Intelligent Processing and Manufacturing Material (IPMM).*, pages 257–267.

[Dasgupta and Forrest, 1995] Dasgupta, D. and Forrest, S. (1995). Tool breakage detection in milling operations using a negative-selection algorithm. Technical Report CS95-05, Dept. of Computer Science,University of New Mexico.

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

[Dasgupta et al., 2002] Dasgupta, D., Majumdar, N., and Nino, F. (2002). Artificial immune systems: A bibliography. Technical report, University of Memphis.

[Davis, 1985] Davis, L. (1985). Job shop scheduling with genetic algorithms. In Grefenstette, J., editor, *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, pages 136–40. Lawrence Erlbaum Associates, Hillsdale New Jersey.

[De Castro and Von Zuben, 2000a] De Castro, L. and Von Zuben, F. (2000a). The clonal selection algorithm with engineering applications. In *GECCO 2002 - Workshop Proceedings*, pages 37–37.

[De Castro and Von Zuben, 2000b] De Castro, L. and Von Zuben, F. (2000b). An evolutionary immune network for data clustering. In *Proceedings of the IEEE Brazilian Symposium on Artificial Neural Networks*, pages 84–89.

[De Castro and Von Zuben, 2001] De Castro, L. and Von Zuben, F. (2001). Ainet: An artificial immune network for data analysis. In Abbass, H., Sarker, R., and Newton, C., editors, *Data Mining: A Heuristic Approach*. Idea Group Publishing, USA.

[Deb and Goldberg, 1989] Deb, K. and Goldberg, D. (1989). An investigation of niche and species formation in genetic function optimization. In Schaffer, J., editor, *Proceedings of the Third International Conference of Genetic Algorithms*, pages 42–50. Morgan Kaufmann.

[DeJong, 1975] DeJong, K. (1975). *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, The University of Michigan, Ann Arbor, MI.

[DeJong, 9898] DeJong, K. (19898). Learning with genetic algorithms. *Machine Learning*, 3:121–138.

[D'haeseleer, 1995] D'haeseleer, P. (1995). Further efficient algorithms for generating antibody strings. Technical Report CS-95-03, The University of New Mexico, Albuquerque.

[Fang et al., 1993] Fang, H.-L., Ross, P., and Corne, D. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems. In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algoriothms*, pages 375–382. San Mateo:Morgan Kaufmann.

[Farmer et al., 1986] Farmer, J., Packard N, H., and Perelson, A. (1986). The immune system, adaption and machine learning. *Physica D*, 22:187–204.

[Forrest et al., 1997a] Forrest, S., Hofmeyr, S., and Somayaji, A. (1997a). Computer immunology. *Communications of the ACM*.

[Forrest et al., 1994] Forrest, S., Javornik, B., Allen, L., and Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA.

[Forrest et al., 1993] Forrest, S., Javornik, B., Smith, R., and Perelson, A. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211.

[Forrest et al., 1997b] Forrest, S., Somayaji, A., and Ackley, D. (1997b). Building diverse computer systems. In *Proceedings of the 6th workshop on Hot Topics in Operating Systems*, Los Alamitos, CA. IEEE Computer Press.

[Fritzke, 1994] Fritzke, B. (1994). Growing cell structures - a self-organizing network for unsuperviesd and supervised learning. *Neural Networks*, 7(9):1441–1460.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems 7*, pages 625–632, Cambridge,MA. MIT Press.

[Fritzke, 1997a] Fritzke, B. (1997a). A self-organising network that can follow non-stationary distributions. In *Proceedings of ICANN-97, International Conference on Artifical Neural Networks*, pages 613–618.

[Fritzke, 1997b] Fritzke, B. (1997b). Unsupervised ontogenic networks. In *Handbook of Neural Computation*, pages C2.1:1–16. IOP Publishing Ltd and Oxford University Press.

[Fukuda et al., 1999] Fukuda, T., Mori, K., and Tsukiyama, M. (1999). *Artificial Immune Systems and their Applications*, chapter Immunity-Based Management Systems, pages 278–288. Springer.

[Gaspar and Collard, 1999] Gaspar, A. and Collard, P. (1999). From gas to artificial immune systems: Improving adaptation in time dependent optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 1859–1866.

[Gaspar and Collard, 2000] Gaspar, A. and Collard, P. (2000). Two models of immunization for time dependent optimization. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*.

[Gathercole and Ross, 1994] Gathercole, C. and Ross, P. (1994). Dynamic training subset selection for supervised learning in genetic programming. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem. Springer-Verlag.

[Gibert and Routen, 1994] Gibert, C. and Routen, T. (1994). Associative memory in an immune based system. In *Proceedings of AAAI-94*, volume 2, pages 852–857. AAAI Press, Menolo Park, California.

[Gilfillan et al., 1993] Gilfillan, S., Dierich, A., Leneur, M., Bonoist, C., and Mathis, D. (1993). Mice lacking tdt: Mature animals with an immature lymphocyte repertoire. *Science*, 261:1175–1178.

[Goldberg and Richardson, 1987] Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J., editor, *Proceedings of the Second International Conference of Genetic Algorithms*, pages 148–154. Morgan Kaufmann.

[Grefenstette, 1984] Grefenstette, J. (1984). Genesis: A system for using genetic search procedures. In *Proceedings of a Conference on Intelligent Systems and Machines*, pages 161–165.

[Hart and Ross, 1998] Hart, E. and Ross, P. (1998). A heuristic combination method for solving job-shop scheduling problems. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN V*, number 1498 in LNCS, pages 845–855. Springer.

[Hart et al., 1998] Hart, E., Ross, P., and Nelson, J. (1998). Solving a real world problem using an evolving, heuristically driven, schedule builder. *Evolutionary Computation*, 6(1):61–81.

[Helman and Forrest, 1994] Helman, P. and Forrest, S. (1994). An efficient algorithm for generating antibody strings. Technical Report CS94-07, Department of Computer Science, University of New Mexico.

[Hely et al., 1997] Hely, T., Willshaw, D., and Hayes, G. (1997). A new approach to kanerva's sparse distributed memory. In *IEEE Transactions on Neural Networks*, pages 101–105.

[Herrmann, 1999] Herrmann, J. (1999). A genetic algorithm for minimax optimzation problems. In Angeline, P., Michalewicz, Z., Schoenhauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1099–1103. IEEE Press.

[Hightower et al., 1995] Hightower, R., Forrest, S., and Perelson, A. (1995). The evolution of emergent organization in immune system gene libraries. In Eshelman, L., editor, *Proceedings of Sixth Annual Conference on Genetic Algorithms*, pages 344–350. Morgan Kaufmann, San Francisco, CA.

[Hofmeyr and Forrest, 2000] Hofmeyr, S. and Forrest, S. (2000). An architecture for an artificial immune system. *Evolutionary Computing*, 8(4):443–473.

[Holland et al., 1986] Holland, J., Holyoak, K., Nisbett, R., and Thagard, P. (1986). *Induction: Processes of inference, learning and discovery*. MIT Press, Cambridge, MA.

[Hunt and Cooke, 1996] Hunt, J. and Cooke, D. (1996). Learning using an artificial immune system. *Journal of Network and Computer Applications*, 19:189–212. Special Issue on Intelligent Systems: Design and Applications.

[Hunt et al., 1995] Hunt, J., Cooke, D., and Holstein, H. (1995). Case memory and retrieval based on the immune system. In Weloso, M. and Aamodt, A., editors, *First International Conference on Case Based Reasoning, (Case-Based Reasoning*

Research and Development, number 1010 in Lecture Notes in Artificial Intelligence, pages 205–216.

[Hunt et al., 1999] Hunt, J., Timmis, J., Cooke, D., Neal, M., and King, C. (1999). *Artificial Immune Systems and Their Applications*, chapter Jisys: The Development on An Immune System for Real World Applications, pages 157–184. Springer-Verlag.

[Ishiguro et al., 1996] Ishiguro, I., Kondo, T., Watanabe, Y., and Uchikawa, Y. (1996). Immunoid: An immunological approach to decentralized behaviour arbitration of autonomous mobile robots. In *Lecture Notes in Computer Science*, volume 1141, pages 666–675.

[Jain et al., 1999] Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

[Jensen and Hansen, 1999] Jensen, M. and Hansen, T. (1999). Robust solutions to job-shop problems. In Angeline, P., Michalewicz, Z., Schoenhauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1138–1144. IEEE Press.

[Jerne, 1973] Jerne, N. (1973). The immune system. *Scientific American*, 229(1):52:60.

[Kanerva, 1988] Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press,Cambridge,MA.

[Knight and Timmis, 2001] Knight, T. and Timmis, J. (2001). Assessing the performance of the resource limited artificial immune system AINE. Technical Report 3-01, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent. CT2 7NF.

[Kohonen, 1982a] Kohonen, T. (1982a). Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

[Kohonen, 1982b] Kohonen, T. (1982b). A simple paradigm for the self-organized formation of structured feature maps. In Amari, S. and Arbib, M. A., editors, *Competition and Cooperation in Neural Nets, Lecture Notes in Biomathematics, Vol. 45*, pages 248–266. Springer, Berlin, Heidelberg. U. S. —Japan Joint Seminar on Competition and Cooperation in Neural Nets, Kyoto, Japan, Feb. 15–19, 1982.

[Leder, 1991] Leder, P. (1991). The genetics of antibody diversity. *Immunology: Recognition and Response*.

[Lin et al., 1997] Lin, S.-C., Goodman, E., and Punch, W. (1997). A genetic algorithm approach to dynamic job-shop scheduling problems. In Bäck, T., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 481–489. Morgan-Kaufmann.

[Marshalls Agriculture, 1998] Marshalls Agriculture, p. (1998). Personal communication.

[Martinetz and Schulten, 1991] Martinetz, T. and Schulten, K. (1991). A 'neural-gas' nework learns topologies. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 397–40.

[Matzinger, 1994a] Matzinger, P. (1994a). Immunological memories are made of this ? *Nature*, 369:605–606.

[Matzinger, 1994b] Matzinger, P. (1994b). Tolerance, danger and the extended family. *Annual Review Immunology*, pages 991–1045.

[Mori et al., 1998] Mori, K., Tsukiyama, M., and Fukuda, T. (1998). Adaptive scheduling system inspired by immune system. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pages 3833–3837.

[Mori et al., 1997] Mori, M., Tsukiyama, M., and Fukuda, T. (1997). Artificial immunity based management system for a semiconductor production line. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pages 851–855.

[Morton and Pentico, 1993] Morton, T. and Pentico, D. (1993). *Heuristic Scheduling Systems*. John Wiley.

[Nakano, 1991] Nakano, R. (1991). Conventional genetic algorithm for job shop problems. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 474–479. Morgan Kaufmann.

[Oprea and Forrest, 1998] Oprea, M. and Forrest, S. (1998). Simulated evolution of antibody gene libraries under pathogen selection. In *Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics*.

[Oprea and Forrest, 1999] Oprea, M. and Forrest, S. (1999). How the immune system generates diversity: Pathogen space coverage with random and evolved antibody libraries. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1651–1656, Orlando, Florida, USA. Morgan Kaufmann.

[Percus et al., 1993] Percus, J., Percus.O, and Perelson, A. (1993). Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. In *Proceedings of the National Academy of Science*, volume 60, pages 1691–1695.

[Perelson, 1989] Perelson, A. (1989). Immune network theory. *Immunological Review*, 10:5–36.

[Perelson et al., 1996] Perelson, A., Hightower, R., and Forrest, S. (1996). Evolution and somatic learning in v-region genes. In *Research in Immunology*, volume 147, pages 202–208.

[Perelson and Oster, 1979] Perelson, A. and Oster, G. (1979). Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self-non-self discrimination. *Journal of Theoretical Biology*, 81:645–670.

[Potter and De Jong, 1998] Potter, M. and De Jong, K. (1998). The coevolution of antibodies for concept learning. In *Parallel Problem Solving From Nature - PPSN V*, pages 530–540. Springer-Verlag.

[Potter and De Jong, 2000] Potter, M. and De Jong, K. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

[Richardson et al., 1989] Richardson, J., Palmer, M., Leipin, G., and Hilliard, M. (1989). Some guidelines for gas with penalty functions. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, pages 191–197. Morgan Kaufmann.

[Roitt et al., 1988] Roitt, I., Brostoff, J., and Male, D. (1988). *Immunology*. Mosby, 5th edition.

[Ross, 2002] Ross, P. (2002). Personal communication.

[Ross et al., 2002] Ross, P., Schulenburg, S., Marín-Blázquez, J., and Hart, E. (2002). Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 942–948, New York. Morgan Kaufmann Publishers.

[Russ et al., 1999] Russ, S., Lambert, A., King, R., Rajan, R., and Reese, D. (1999). An artificial immune system model for task allocation. In *Proceedings of the Symposium on High Performance Distributed Computing*.

[Sjödin, 1996] Sjödin, G. (1996). Getting more information out of sdm. In von der Malsburg, C., von Seelen, W., Vorbruggen, J., and Sendhoff, B., editors, *Proceedings of ICANN 96*, pages 477–482. Springer.

[Smith and Coit, 1997] Smith, A. and Coit, D. (1997). *Handbook of Evolutionary Computing*, chapter Constraint Handling Techniques - Penalty Functions. Oxford University Press. Chapter C5.2.

[Smith et al., 1996] Smith, D., Forrest, S., and Perelson, A. (1996). Immunological memory is associative. In *Workshop Notes, Workshop 4: Immunity Based Systems, International Conference on Multiagent Systems*, pages 62–70. Kyoto, Japan.

[Smith et al., 1999] Smith, D., Forrest, S., and Perelson, A. (1999). *Artificial Immune Systems and Their Applications*, chapter Immunological Memory is Associative, pages 105–112. Springer-Verlag.

[Smith et al., 1993] Smith, R., Forrest, S., and Perelson, A. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149.

[Stadnyk, 1987] Stadnyk, I. (1987). Schema recombination in pattern recognition problems. In who, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, page ?, Hillside, NJ. Lawrence Erlbaum Assoc.

[Tew and Mandel, 1979] Tew, J. and Mandel, T. (1979). Prolonged antigen half-life in the lymphoid follicules of antigen-specifically immunized mice. *Immunology*, 37:69–76.

[Timmis, 2000a] Timmis, J. (2000a). On parameter adjustment of the immune inspired machine learning algorithm AINE. Technical Report 12-00, Computing Laboratory, Univeristy of Kent at Canterbury, Canterbury, Kent. CT2 7NF.

[Timmis, 2000b] Timmis, J. (2000b). Visualising artificial immune networks. Technical Report UWA-DCS-00-034, University of Wales, Aberystwyth.

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

[Timmis et al., 1999] Timmis, J., Neal, M., and Hunt, J. (1999). Data analysis using artificial immune systems, cluster analysis and kohonen networks: Some comparisons. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 922–927. IEEE.

[Timmis et al., 2000] Timmis, J., Neal, M., and Hunt, J. (2000). An artificial immune system for data analysis. *Biosystems*, 55(1/3):143–150.

[Tonegawa, 1983] Tonegawa, S. (1983). Somatic generation of antibody diversity. *Nature*, 302:575–581.

[Varela and Coutinho, 1988] Varela, F. and Coutinho, A. (1988). Cognitive networks: Immune,neural and otherwise. *Theoretical Immunology*, 2:359–371.

[Weigert et al., 1970] Weigert, M., Cesari, I., Yonkovitch, S., and Cohn, M. (1970). Variability in the light chain sequences of mouse antibody. *Nature*, 228:1045=1047.

[Weinand, 1990] Weinand, R. (1990). Somatic mutation, affinity maturation and antibody repertoire: A computer model. *Journal of Theoretical Biology*, 143(3):343–382.

[Whitely and Starkweather, 1990] Whitely, D. and Starkweather, T. (1990). Genitor ii: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(3):189–214.

[Wu et al., 1999] Wu, S., Byeon, E., and Storer, R. (1999). A graph-theoretic decomposition of the job-shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47.