CHAPTER 6

IMMUNOCOMPUTING

"Like the central nervous systems, the immune system is individualized. Identical twins, born with identical DNA, develop different immune systems, just as they develop different brains. Each person's immune system records a unique history of individual life, because ... the immune system, like the brain, organizes itself through experience. Thus the brain and the immune system establish individuality at two levels: they help us adapt to life and so preserve us, and they make a record of what has happened."

(I. R. Cohen, Tending Adam's Garden: Evolving the Cognitive Immune Self, Academic Press, 2004; p. 5)

"There is probably no system, living or manmade, that can protect itself as effectively, on so many levels, and from so many different diseases and infections as the human immune system."

(N. Forbes, Imitation of Life: How Biology is Inspiring Computing, MIT Press, 2004; p. 98)

6.1 INTRODUCTION

The immune system of vertebrates is an intricate collection of distributed cells, molecules, and organs that altogether play the important role of maintaining a dynamic internal state of equilibrium in our bodies. Its complexity has been compared to that of the brain's in many respects: immune systems are capable of recognizing foreign and internal signals; controlling the action of immune components; influencing the behavior of other systems, such as the nervous and the endocrine systems; and, most importantly, learning how to fight against disease causing agents and extracting information from them.

Immunocomputing is a terminology introduced to describe a new computational approach that aims at implementing information processing principles from proteins and immune networks in new kinds of computer algorithms and software, leading to the concept of an immunocomputer (Tarakanov et al., 2003). *Artificial immune systems* (AIS)[1], by contrast, is a terminology that refers to adaptive systems inspired by theoretical and experimental immunology with the goal of solving problems (de Castro and Timmis, 2002). They encompass any system or computational tool that extracts ideas and metaphors from the biological immune system in order to solve problems.

Together with swarm intelligence, AIS constitute one of the youngest fields of investigation in nature-inspired computing. Despite its youth, the bibliography on AIS is becoming vast. Added to the success of many applications, the creati-

[1] Although AIS and immunocomputing mean slightly different approaches, in this chapter the terminology immunocomputing will be used as a synonym for artificial immune systems.

on of the series of International Conferences on Artificial Immune Systems (ICARIS) in 2002 (Timmis and Bentley, 2002; Timmis et al., 2003; Nicosia et al., 2004; Jacob et al., 2005), in addition to the many special sessions and tracks on the major evolutionary computation conferences, helped to promote the field and attract more researchers.

This chapter provides a basic introduction to artificial immune systems. It starts with a review of some immunological background necessary for the development and understanding of AIS, and then follows with the description of a framework to design artificial immune systems. The focus is on four important immunological principles and theories: pattern recognition within the immune system; adaptive immune responses; self/nonself discrimination; and the immune network theory. The framework to design an AIS introduces a generic structure to create abstract models of immune cells and molecules, presents some measures to quantify the interactions of these elements with each other and the environment, and describes some general-purpose immune algorithms. Although most sections describe portions of the immune system, Section 6.2.7 on the mammalian immune system presents a slightly broader view of the immune system to illustrate part of its complexity and the various types of cells and molecules involved in an immune response.

6.2 THE IMMUNE SYSTEM

All living beings have the ability to present resistance to disease-causing agents, known as *pathogens*. These include viruses, bacteria, fungi, and parasites. The nature of this resistance varies from one species to the other, and is a function of the complexity of the organism. Mammals, in particular human beings, have developed a highly sophisticated immune system that acts together with several other bodily systems (such as the nervous and the endocrine system) to maintain life. The primary role of the immune system is to protect our bodies against infections caused by pathogens (Janeway et al., 1999; Tizard, 1995).

The defense must occur in many levels and has to cover the whole body. Therefore, various levels of defense mechanisms and barriers have evolved in order to result in sufficient protection. For instance, physical barriers such as the skin, and biochemical barriers such as the pH levels and the saliva, are sometimes considered as parts of the immune system. Apart from these physical and biochemical barriers, the immune system can be divided into *innate immune system* and *adaptive immune system*, composed of diverse sets of cells, molecules and organs that work in concert to protect the organism.

The innate immune system is very important as a first line of defense against several types of pathogens and is also crucial for the regulation of the adaptive immune system. Cells belonging to the innate immune system are capable of recognizing generic molecular patterns (a type of molecular signature) that are only present in pathogens, and can never be found in the cells of the host. Once a pathogen has been recognized by a cell of the innate immune system, this cell signals (through chemical messengers) other immune cells, including those of

the adaptive immune system, to start fighting against the pathogen. Therefore, the innate immune system plays a major role in providing *co-stimulatory* signals for the adaptive immune system. Co-stimulatory signals are usually provided by the innate immune system when the organism is being damaged in some way, such as when cells are being killed by viruses. For the most types of pathogens, the adaptive immune system cannot act without the co-stimulatory signals provided by the innate immune system.

Nevertheless, not all pathogens can be recognized by the innate system. Some specific pathogens are only recognized by cells and molecules of the adaptive immune system, also called specific immune system. The adaptive immune system possesses some particular features that are important from a biological and computational perspective. For instance, it can adapt to those molecular patterns previously seen and it generates and maintains a stable memory of known patterns.

After a certain pathogen has been eliminated by the adaptive system, the innate system plays a role in signaling the adaptive system that the foreign agent has been defeated. Another way the innate immunity is important for the adaptive immunity is in that the latter usually requires some time before it starts acting. Thus, the innate immune system tries to get the pathogen at bay until the adaptive immune system can act, but the innate system by itself is usually not capable of removing the infection.

Once the adaptive immune system is prepared to act, it can adapt to the invading pathogen and create specific molecular patterns to fight against the same or a similar future infection of this type. This adaptive (evolutionary and learning) capability has been explored by immunologists and medics for over two centuries in order to protect us against known pathogens. The vaccination principle is very much embodied in this learning capability of the immune system. By inoculating a healthy individual with weakened or dead samples of some specific disease-causing agents (pathogens), the immune system is allowed to generate sets of molecular structures specific in recognizing and fighting against that pathogen, without subjecting the organism to the unpleasant symptoms of the disease. The mechanisms underlying this adaptability of the immune system have also been broadly explored in immunocomputing, and these will be discussed later.

Last, but not least, there are theories that suggest the immune system is a dynamic system whose cells and molecules are capable of interacting with each other. This viewpoint establishes the idea that pathogens are responsible for modifying the structure of a dynamic immune system, while the other more traditional perspectives suggest that the immune system is composed of discrete sets of cells and molecules that are only activated by pathogens. This section reviews some basic immune theories and principles that have been used for the design and application of artificial immune systems, including the more controversial *immune network theory*. The adaptive immune system is emphasized due to its adaptation, learning, and memory capabilities.

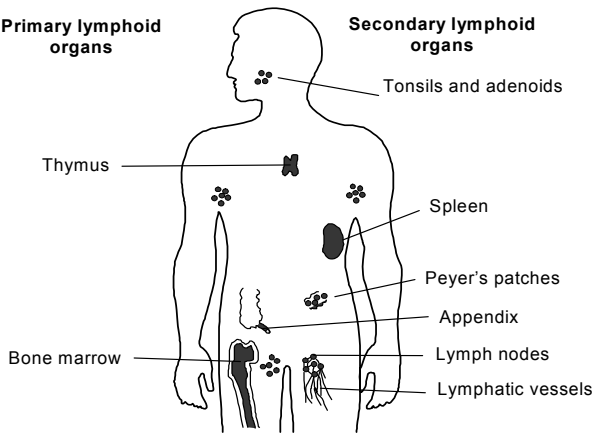


Figure 6.1: Physiology of the immune system. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

6.2.1. Physiology and Main Components

One remarkable feature of the immune system is its distributivity all over the body. There are cells, molecules, and organs in all parts of the organism, as illustrated in Figure 6.1. The organs composing the immune system, sometimes referred to as the *lymphoid system*, are called *lymphoid organs*. The *primary lymphoid organs* are those responsible for the production, growth, development, and maturation of *lymphocytes*, which are specialized white blood cells that bear specific receptors to recognize pathogens. The *secondary lymphoid organs* are the sites where the lymphocytes interact with pathogens.

The two main types of lymphocytes are the *B-cells* and the *T-cells*, or *B-lymphocytes* and *T-lymphocytes*. The B-cells are so named because they maturate (i.e., become able of fighting against diseases) in the bone marrow, and the T-cells maturate in the thymus. Both present several important features from the biological and computational perspectives. They have surface receptor molecules capable of recognizing specific molecular patterns present on pathogens. They are also capable of multiplying themselves through a *cloning* process, and the B-cells are capable of suffering genetic variation during reproduction. Therefore, these cells have their number varied with time and B-cells can have their molecular structure changed as well.

The two primary lymphoid organs, bone marrow and thymus, have important functions in the production and maturation of immune cells. All blood cells are generated within the bone marrow, and the B-cells also maturate - become ready to act as an immune cell - within the bone marrow. T-cells are generated within the bone marrow but migrate to the thymus where they will eventually become immunocompetent cells. The thymus is also believed to be the site where the T-cells learn to recognize self, in a process known as negative selection of T-cells (Section 6.2.4).

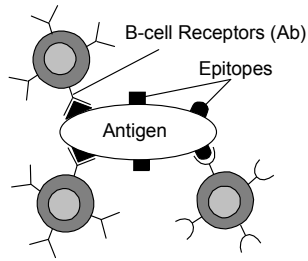


Figure 6.2: Immune cells, in particular B-cells, with their molecular receptors detached recognizing the epitopes on antigens. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

6.2.2. Pattern Recognition and Binding

The first step in promoting an *immune response*, that is, a response of the immune system against infection, is the recognition of the pathogen. Those portions of the pathogens that can promote an adaptive immune response are called *antigens*. Antigens present surface molecules, named *epitopes*, that allow them to be recognized by immune cells and molecules. Figure 6.2 depicts immune cells, in particular B-cells, and their receptors recognizing the epitopes of a given antigen. Note that B-cells are believed to be monospecific, i.e., present a single type of receptor on the surface. On the contrary, antigens can present various patterns of epitopes.

There are several types of immune cells and molecules. Most AIS focus on the adaptive immune system with emphasis either on B-cells or T-cells, which are the most important cells from the adaptive immune system. B-cells also have been the focus of most research on AIS because they are capable of altering their genetic composition during an immune response. The B-cell receptor is also known as *antibody* or *immunoglobulin*.

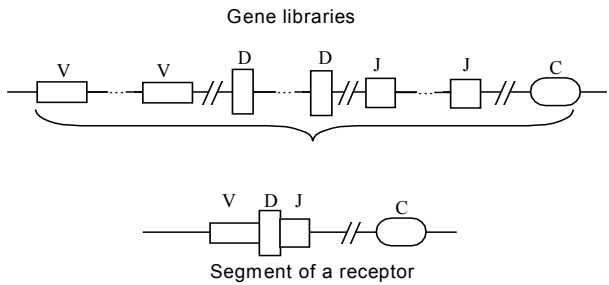


Figure 6.3: Gene rearrangement process that results in the production of part of the receptor molecule of a B-cell receptor. Segments from different libraries (V, D, J, and C) are joined together to form the antibody.

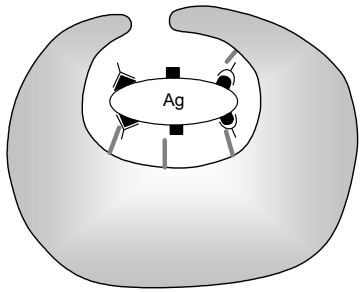


Figure 6.4: A phagocyte recognizes a pathogen covered with antibodies and thus destroys the whole complex.

The receptor molecules of both B-cells and T-cells are generated by the recombination of gene segments concatenated from several gene libraries, what allows the immune system to produce a large variety of receptors from a finite genome. The gene rearrangement process that results in the production of parts of a B-cell receptor molecule is illustrated in Figure 6.3. In this process, some genes are taken from the libraries (e.g., one gene from each of the libraries: V, D, J, and C) and concatenated so as to form the receptor molecule.

In order for the receptor on an immune cell to recognize an antigen, some portion of the antigen - the epitope - has to have a shape complementary to the shape of the receptor molecule of the immune cell (see Figure 6.2). However, these shapes do not have to be fully complementary for a recognition event to occur, a partial recognition (or match) between both molecules - receptor and epitope - is sufficient. The degree of recognition or interaction between these two molecules is termed *affinity*; the better the recognition, the higher the affinity, and vice-versa. Affinity is a very important concept in artificial immune system. Note that recognition is not only based on shape complementarity, it also depends upon a number of chemical cues, van der Waals interactions, etc.

It is important to have in mind that recognition by itself is not sufficient to eliminate an infectious agent; there must be a binding between the cell receptor and the antigen so as to signal other immune cells, mainly those of the innate immune system, to destroy the complex formed by the cell receptor bound with the antigen. Figure 6.4 illustrates the case of an antigen covered with antibodies (Y-shaped molecules attached to the antigen Ag). The complex formed by the antigen covered with antibodies signals other immune cells, in this case, big cell eaters known as *phagocytes*, to ingest and digest the complex, and thus destroy the antigen.

6.2.3. Adaptive Immune Response

Several theories were proposed as attempts to explain how the immune system copes with antigens. For instance, L. Pasteur suggested, in the late 1800s, that elements contained in the vaccine were capable of removing nutrients essential

to the body and, thus, avoiding the growth and proliferation of pathogens. It were M. Burnet and D. Talmage who in the mid-1900s proposed and formalized the *clonal selection theory* of adaptive immunity (Burnet, 1959), broadly accepted as an explanation of how the adaptive immune system responds to pathogens. Together with the theory of *affinity maturation* of antibodies (Nossal, 1993; Storb, 1998), clonal selection forms the core of an adaptive immune response, and both have been used in the literature of artificial immune systems to design adaptive systems for problem solving.

According to the clonal selection theory, the immune system is composed of sets of discrete cells and molecules that remain at rest until a pathogenic agent invades the organism. After invasion, some subset of these immune cells, either B-cells or T-cells, is capable of recognizing the invading antigen and binding with it. This recognition process stimulates the immune cells capable of recognizing the antigen to start reproducing themselves. Reproduction in this case is asexual; it refers to cellular reproduction, a process of *mitotic* cell division, and is sometimes called *cloning* or *clonal expansion* (Figure 6.5). Thus, a subset of cells (*clone*) capable of recognizing a specific type of antigen is generated.

As with all reproductive events, the B-cell division (or reproduction) process may be subject to an error (*mutation*). A particular feature of the immune system is that mutation occurs in proportion to the cell affinity with the antigen that the

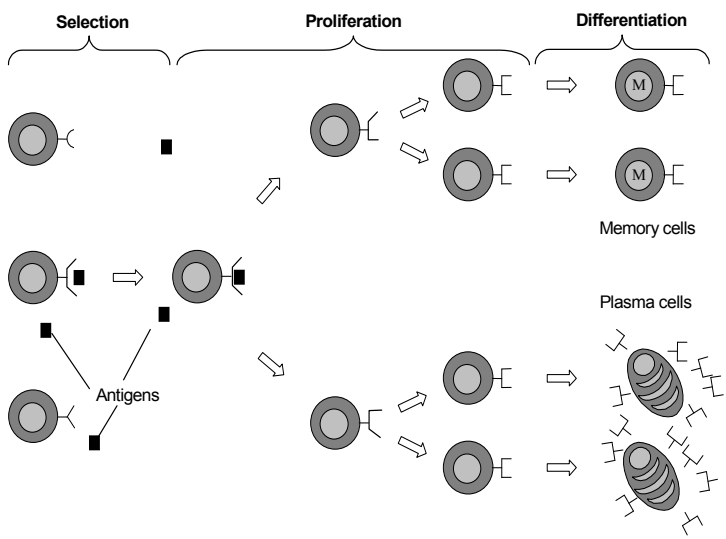


Figure 6.5: Clonal selection, expansion, and affinity maturation. From the repertoire of B-cells, the one(s) presenting higher affinities with the antigen are selected and stimulated to proliferate. During each proliferation stage, mutation may occur, thus allowing the cell receptor to become more adapted to the antigen presented. Those mutated immune cells with higher affinities with the antigen are selected to become memory and antibody secretor (plasma) cells.

parent cell bind. If the affinity between antigen (Ag) and antibody (Ab) is high, then the mutation rate is low; and if the Ag-Ab affinity is low, then high mutation rates are allowed. This is very interesting because it suggests that mutation in the immune system is regulated by the degree of recognition of an antigen by an antibody (Allen et al., 1987). In addition, there are authors (e.g., Kepler and Perelson, 1993) who suggest that a short burst of mutation is followed by periods of low mutation rates to allow for selection and clonal expansion (reproduction).

Another aspect of clonal selection is that it is local. As the immune system is distributed throughout the organism, only those cells located closer to the site of infection are recruited for combat. More cells from the neighborhood can be recruited, but not all immune cells are involved in an adaptive immune response. Figure 6.5 summarizes the clonal selection and affinity maturation processes. The B-cell(s) with higher affinity with the antigen are selected to proliferate. During mitosis, the cells can suffer somatic mutation, and a selection mechanism allows those mutated progenies with increased affinities to be retained in a pool of memory cells. Memory cells usually have very long life spans, on the order of years, even decades.

Adaptation via Clonal Selection

Clonal selection and affinity maturation suggest that the immune system becomes more adapted to the environmental stimulation provided by pathogens. Two degrees of plasticity (forms of adaptation) can be observed in these processes: structural and parametric plasticity. The immune cells successful in recognizing and binding with antigens are selected to proliferate, while those cells that do not perform any role whatsoever in an immune response die, and are eliminated from the repertoire of immune cells. Affinity maturation leads to a parametric adaptation in the molecular structure of antibodies, and thus contributes to the adaptability of the immune system, and the learning of molecular patterns present on pathogens.

The idea can be likened to that of a reinforcement learning strategy (Section 4.3.3), as illustrated in Figure 6.6. When an antigen Ag_1 invades the organism, a few specific antibodies capable of recognizing and binding with this antigen will be selected to proliferate. But some time is required until the immune cells start reproducing themselves so as to fight against Ag_1. After this *lag phase*, sufficient antibodies that recognize and bind with Ag_1 are generated so as to guarantee the elimination of the antigen. In a future or secondary exposition to this same antigen Ag_1, a faster and stronger response is promoted by the immune system. If a new antigen Ag_2 is presented to the immune system, then a pattern of response similar to that to Ag_1 occurs, in the sense that the primary response to Ag_2 is also slow and with a long lag phase. This is because the immune response is specific, meaning that antibodies successful in recognizing a given antigen are specific in recognizing that antigen or a similar one Ag_{11}. Therefore, the response of the antibodies initially primed with Ag_1 to a similar antigen Ag_{11} is similar to the secondary response to Ag_1.

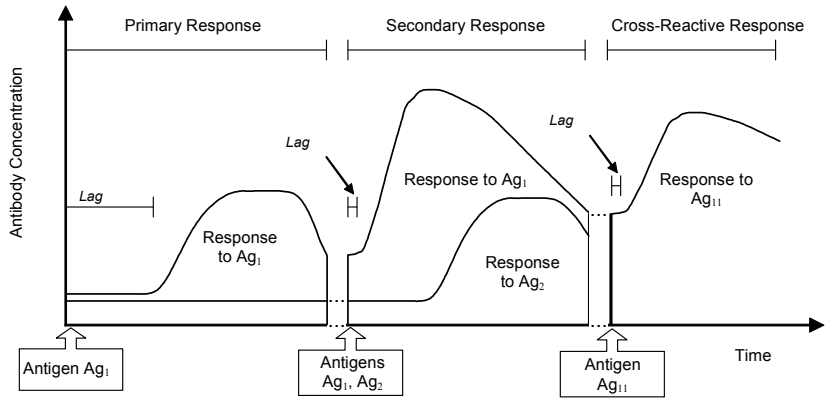


Figure 6.6: Primary, secondary and cross-reactive immune responses. After an antigen Ag_1 is seen once (primary response), subsequent encounters with the same antigen (secondary response), or a similar one Ag_1' (cross-reaction), will promote a faster and more effective response not only to Ag_1, but also to Ag_1'. Primary responses to antigens Ag_1 and Ag_2 are qualitatively equivalent. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

In the (artificial) immune system literature this is known as *cross-reactive response*, and can be likened to the generalization capability of neural networks. While discussing neurocomputing, we argued that generalization is important for the creation of models of the world. In the present context, cross-reactivity is also important for the creation of models of the antigenic universe. If you were not able of learning different types (e.g., Ag_1 and Ag_2) of molecular structures, then death from infection would be inevitable. Of course if the variation between these patterns is too great, cross-reactivity may not be enough for a successful secondary response and thus, a primary response has to be raised.

Thus, adaptation and learning in the immune system involve the variation in the population sizes and affinity of specific cells successful in recognizing and binding with pathogens. The primary adaptive responses may take some time to act, but the secondary response is fast and efficient. Generalization or cross-reactivity is also present in the adaptive immune response.

Clonal Selection and Darwinian Evolution

Chapter 3 presented the neo-Darwinian theory of evolution. Basically, it states that an evolutionary procedure can be defined as the one involving a population of individuals capable of reproducing, and subjected to genetic variation followed by natural selection. By looking at the clonal selection and affinity maturation principles of an adaptive immune response it is easy to note that clonal selection is a type of evolutionary process. It may come as no surprise, thus, the fact that clonal selection immune algorithms may represent a new type of evolutionary algorithm that borrows ideas from the immune system.

In (de Castro and Timmis, 2002), the authors suggested that natural selection can act on the immune system at two distinct levels. First, clonal selection and affinity maturation can be seen as a *microevolutionary process*, i.e., one that occurs within organisms in a time scale orders of magnitude smaller than that of the species (organism) evolution. Second, the immune system contributes to natural selection in a sense that individuals with malfunctioning immune systems have lesser probabilities of survival and thus reproduction. Finally, it was also suggested that the immune system is capable of generating and maintaining diverse sets of cells and molecules that cover the various niches of antigens presented to the immune system.

6.2.4. Self/Nonself Discrimination

One question that for long has intrigued scientists in various fields is that of how the immune system differentiates between the cells of the organism, known as *self*, and the foreign elements capable of causing disease, known as *nonself*. There are various theories that try to approach this question, and one of these involves the *negative selection of T-cells* within the thymus. Other less orthodox proposals are the idea that the immune system evolved to discriminate between infectious nonself and noninfectious self (Janeway, 1992), and the Danger theory that suggests the immune system is capable of recognizing stress or damage signals (Matzinger, 1994).

When an immune cell encounters an antigen, several outcomes might arise. For instance, it has been discussed that if the antigen is nonself, i.e., disease-causing, then the clonal expansion of those cells successful in recognizing and binding with the antigen will occur. But this is not the whole picture. In the case an antigen is recognized by an immune cell while it is patrolling the organism for nonself, a *second signal*, also called *co-stimulatory signal*, from other immune cells is required before an adaptive immune response can be launched.

What if the antigen is a self-antigen? There are a few possibilities in this case and only the negative selection of T-cells within the thymus will be considered here. In a simplified form, if a self antigen is recognized by an immature T-cell within the thymus, this cell is purged from the repertoire of T-cells, else it becomes an immunocompetent cell and is released to circulate throughout the body in the search for nonself antigens. This process, called *thymic negative selection of T-cells* or simply *negative selection*, is only possible because the thymus is protected by a blood-thymic barrier that filters out any molecule that does not belong to self. Thus, all molecules within the thymus are self molecules, and the immature T-cells learn to be *tolerant* (not respond to) to the self molecules while within the thymus.

6.2.5. The Immune Network Theory

The last immune theory to be reviewed in this chapter was formalized by N. Jerne in the mid 1970s. In contrast to the clonal selection theory that proposes an immune system composed of discrete cells and molecules at rest and stimulated by nonself antigens, the *immune network theory* suggests that all immune cells

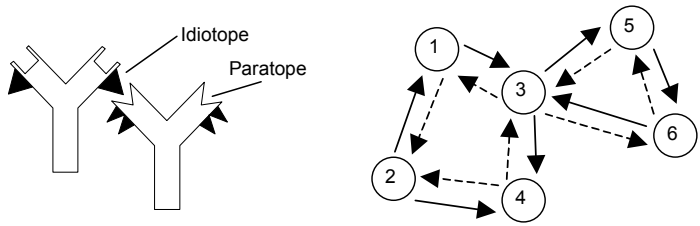


Figure 6.7: The immune network theory. (a) The recognition between two antibodies. The idiotope of an antibody is recognized by the complementary paratope of another antibody. (b) In the network depicted, cell 1 recognizes cell 3 and is thus stimulated, while cell 3 is suppressed; cell 2 is stimulated by the recognition of cells 1 and 4, which by themselves are suppressed by cell 2; and so forth. Solid arrow: stimulation. Dashed arrow: suppression.

and molecules recognize each other (Jerne, 1974). Thus, no foreign stimulation is necessary for a working (dynamic) immune system.

Jerne suggested that in and around the receptors of an immune cell there are molecular patterns, called *idiotopes*, that allow this cell to be recognized by other cells and molecules of the immune system. The idiotope of an immune cell can be recognized by a complementary paratope of another immune cell, as illustrated in Figure 6.7(a). This view suggests a dynamic immune system with behavior governed by the patterns of interactions of immune cells and molecules, and whose state is changed due to the perturbation caused by nonself antigens. As the immune cells are assumed to be monospecific, the recognition of an immune cell can be viewed as corresponding to the recognition of its receptor. When the receptor on one cell recognizes the receptor on another cell, the recognizing cell is stimulated, and the cell recognized is suppressed, as illustrated in Figure 6.7(b). The effects of stimulation and suppression vary from one model to the other, and will be discussed further.

Adaptation and Learning via Immune Network

To formally describe the immune network activity, F. Varela and collaborators (Varela et al., 1988) introduced three important network concepts: *structure*, *dynamics*, and *metadynamics*:

1. *Structure*: as with the case of neural networks, the immune network structure corresponds to the patterns of connectivity between the cells and molecules in the network, i.e., how the network components are linked together. However, immune networks do not usually have predefined architectures. Instead, the immune network structure tends to follow that of the antigenic environment and the molecular structure of the immune cells.
2. *Dynamics*: the immune network dynamics corresponds to how the network cells and molecules vary with time. This is one of the main features

that allow immune networks to adapt to the environment, what is a consequence of the interactions between the components of the network themselves and with the environment.

3. *Metadynamics*: one important feature of the immune system is the constant generation of immune cells and the death of unstimulated cells. New cells are constantly being generated and replacing cells that do not play any role in immune responses. The generation and death of immune cells results in the creation and elimination of network connections as well. This capability of having its structure varied with time is also referred to as the double plasticity of the immune system.

6.2.6. Danger Theory

With a conceptually different viewpoint, P. Matzinger (1994, 2002) introduced what came to be known as the *danger theory*. In essence, the danger model adds another layer of cells and signals to the self/nonself discrimination models. It proposes that antigen-presenting cells (APCs), such as those exposed to pathogens, toxins, mechanical damage, etc., are activated by the alarms caused by these phenomena.

The danger model tries to answer one of the fundamental questions in immunology: How is self-tolerance induced? It suggests that the immune system is more concerned with damage (preventing destruction) than with foreignness. It takes into account issues like what happens when bodies change (e.g., through puberty, pregnancy, aging, etc.); why are there T- and B-cells specific for self-antigens; why do we not mount immune responses to vaccines; why neonates are easily tolerizable; why silicone, well boiled bone fragments, or solitary haptens do not elicit immune responses; why do we fail to reject tumors; and so forth (Matzinger 1994, 2002).

A puzzling question is how to distinguish between dangerous and non-dangerous. At the same time there are several foreign things that are dangerous, such as bacterial toxins, viruses, worms, and others, there are also dangerous 'self', such as tumors, and nondangerous foreign, such as beneficial bacteria and nonlytic viruses. The new danger theory proposes that antigen presenting cells are activated by danger/alarm signals from injured cells, such as those exposed to pathogens, toxins, and mechanical damage. The mechanisms of immune activation would be a consequence of cell/tissue damage. Cell death is not always a result of parasitic attack. It is a normal event during embryonic development, formation and death of hematopoietic cells, maturation and expulsion of oocytes, etc. In such cases, death is controlled, usually apoptotic, and cells that die by these normal programmed processes, are usually scavenged before they desintegrate. By contrast, cells dying by stress or necrotically release their contents in the surroundings and these serve as (danger) signals, as illustrated in Figure 6.8. The key issue is that danger signals are not sent by normal (healthy) cells, only by injured tissues. The danger signals can be active or passive (Matzinger, 1994). Abrupt changes in the condition of a cell, like, for instance, temperature variation or infection, elaborate a series of heat chock proteins that

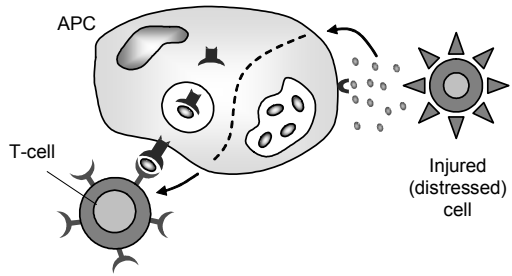


Figure 6.8: When a cell is injured (or stressed) it produces some damage (or stress) signals that are recognized by cells of the immune system.

aid their recovery, and serve as danger signals. Internal molecules, normally not secreted, may also serve as a danger signal; thus, any cell damage caused by a cut, bruise and infection, can be noted.

6.2.7. A Broader Picture

Although this chapter has reviewed a number of theories and principles that try to explain how the immune system behaves, the overall system is much more complex than this chapter could cover. To provide the reader with a reasonably broader picture of the basic recognition and activation mechanisms involved in an immune response, consider the illustration presented in Figure 6.9 (de Castro and Timmis, 2002).

Part (I) of Figure 6.9 shows how specialized *antigen presenting cells*, called APCs, such as phagocytes, circulate throughout the body ingesting and digesting antigens. These antigens are fragmented into *antigenic peptides* (Nossal, 1993). Part of these peptides bind with some molecules termed MHC and the whole complex is presented in the APC cell surface (II). The T-cells carry surface receptors that allow them to recognize different MHC/peptide complexes (III). Once activated by the MHC/peptide recognition, the T-cells become activated, divide, and secrete chemical signals that stimulate other components of the immune system to enter into action (IV).

In contrast to the T-cells, B-cells have receptors with the ability to recognize parts of the antigens free in solution (V). The surface receptors on these B-cells respond to a specific antigen. When a signal is received by these B-cell receptors, the B-cell is activated and will proliferate and differentiate into *plasma cells* that secrete antibody molecules in high volumes (VI). These released antibodies (which are soluble forms of the B-cell receptors) are used to neutralize the pathogen (VII), leading to their destruction. Some of these activated B- and T-cells will differentiate into *memory cells* that remain circulating through the organism for long periods of time, thus guaranteeing future protection against the same (or a similar) antigen that elicited the immune response.

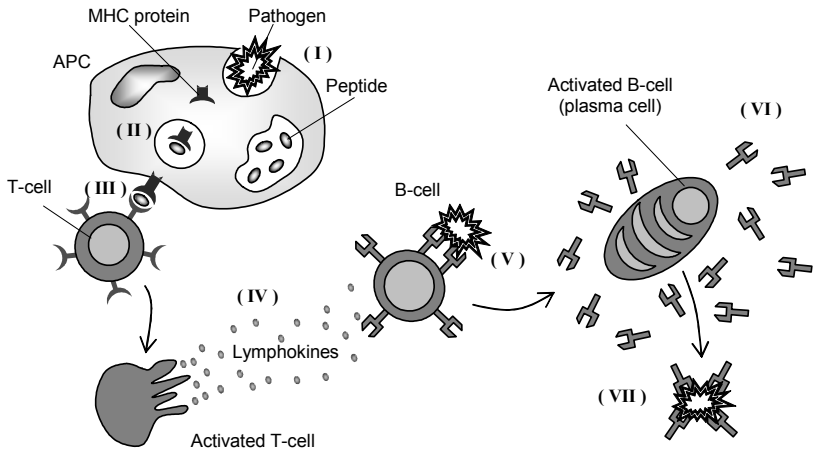


Figure 6.9: Broader picture of the basic immune recognition and activation mechanisms. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

6.3 ARTIFICIAL IMMUNE SYSTEMS

Differently from evolutionary algorithms that emerged from one main central idea and developed into several branches and variations, artificial immune systems were proposed taking into account various different features, processes, and models of the immune system. The number of applications and algorithms present in the literature of AIS is vast, but some core ideas have been broadly explored, namely, clonal selection and affinity maturation, negative selection, and immune networks. Many new proposals involving mainly concepts from innate immunity and the danger theory have appeared in the last ICARIS conferences (Timmis et al., 2003; Nicosia et al., 2004; Jacob et al., 2005), but still not with a common ground among them.

To design artificial immune systems, de Castro and Timmis (2002) have proposed a layered approach based on the *immune engineering* framework introduced by de Castro (2001, 2002, 2003). The term *immune engineering* refers to " … a meta-synthesis process that is going to define the tool for solving a given problem based upon the features of the problem itself, and then apply it to obtain the solution to the problem. Instead of trying to create accurate models of the immune system, the immune engineering must try to develop and implement pragmatic models inspired by the immune system. These must preserve some of the essential properties of the immune system which demonstrate to be implementable and efficient for the development of engineering tools." (de Castro, 2001; p. 44). Note that the distinction between this definition and that of artificial immune systems is sharp; immune engineering is basically concerned with the problem of designing AIS.

The immune engineering process that leads to a framework to design AIS is composed of the following basic elements (de Castro and Timmis, 2002):

- A *representation* for the components of the system.
- A set of *mechanisms to evaluate the interaction* of individuals with the environment and each other. The environment is usually simulated by a set of input stimuli or patterns, one or more fitness function(s), or other means.
- *Procedures of adaptation* that govern the dynamics and metadynamics of the system, i.e., how its behavior varies over time.

6.3.1. Representation

There are several types of immune cells and molecules composing the immune system. The first step toward designing an artificial immune system is to devise a scheme to create abstract models of these immune cells and molecules. Any immune response requires the recognition of an antigen by a cell receptor. Recognition in the immune system occurs mainly through shape complementarity between these two molecules: the cell receptor and the antigen. Note that some cell receptors (e.g., antibodies) can be released from the cell surface, and can thus be found free in solution.

To model this shape recognition process in the immune system, A. Perelson and G. Oster (1979) introduced the concept of *shape-space*. The shape-space approach assumes that all the properties of receptor molecules relevant to determining their interactions with each other and foreign antigens can be described by a *data structure*. Thus, the data structure, which usually depends upon the problem being studied, corresponds to the *generalized shape* of a molecule and is sufficient to quantify the affinity between molecules.

Although recognition within the immune system occurs through shape complementarity, most AIS quantify the degree of similarity instead of complementarity or dissimilarity between data structures. Thus, it is important to known if shape complementarity or similarity is going to be the target of recognition in the AIS developed. The most common type of data structure is an *attribute string*, which can be a real-valued vector, an integer string, a binary string, or a symbolic string. These result in diverse shape-spaces:

- *Real-valued shape-space*: the attribute strings are real-valued vectors.
- *Integer shape-space*: the strings are integers.
- *Hamming shape-space*: the strings are built out of a finite alphabet of length k.
- *Symbolic shape-space*: usually composed of different types of attribute strings where at least one of them is symbolic, such as ‘age’, ‘height’, etc.

Assume the case in which any receptor molecule of an immune cell is called an antibody and is represented by the set of coordinates $\mathbf{Ab} = \langle Ab_1, Ab_2, ..., Ab_L \rangle$; an antigen is given by $\mathbf{Ag} = \langle Ag_1, Ag_2, ... , Ag_L \rangle$, where boldface letters correspond to attribute strings. Without loss of generality, antigens and antibodies are assumed to be of same length L. The interaction of antibodies, or of an antibody

and an antigen, can be evaluated via a *distance* or a *similarity measure*, also termed *affinity measure*, between their corresponding attribute strings, depending if shape complementarity or similarity is being sought. Thus, the affinity measure performs a mapping from the interaction between two attribute strings into a nonnegative real number that corresponds to their *affinity* or *degree of match*, $S^L \times S^L \rightarrow \Re^+$. In the case recognition is proportional to shape complementarity, the higher the distance (for instance, for Hamming shape-spaces) the better the recognition. By contrast, if recognition is proportional to similarity, the smaller the distance, the better the recognition.

6.3.2. Evaluating Interactions

Assuming the various types of attribute strings to represent the generalized shapes of molecules in the immune system, each one of these types will require a particular class of *affinity measure*. Real-valued shape-spaces require affinity measures that deal with real-valued vectors; Integer shape-spaces require affinity measures that deal with integer strings; and so forth. For real-valued shape-spaces, the most common affinity measures are the Euclidean and Manhattan distances, given by Equation (6.1) and Equation (6.2), respectively:

$$D = \sqrt{\sum_{i=1}^{L} (Ab_i - Ag_i)^2} \tag{6.1}$$

$$D = \sum_{i=1}^{L} \left| Ab_i - Ag_i \right| \tag{6.2}$$

For Hamming shape-spaces, the Hamming distance can be used to evaluate the affinity between two cells. In this case, the molecules are represented as sequences of symbols over a finite alphabet of length k. Equation (6.3) depicts the Hamming distance used to evaluate the affinity between two attribute strings of length L in a Hamming shape-space.

$$D = \sum_{i=1}^{L} \delta_i, \text{where} \quad \delta_i = \begin{cases} 1 & \text{if } Ab_i \neq Ag_i \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

If binary strings, termed *bitstrings*, $k \in \{0,1\}$, are used to represent the molecules, then one has a *binary Hamming shape-space*, or a *binary shape-space*. If ternary strings, i.e., $k = 3$, are used to represent the molecules, then one has a *ternary Hamming shape-space*, or *ternary shape-space*; and so on.

To illustrate the use of the Hamming distance to evaluate affinity, consider two arbitrary strings **Ag** = [1 0 0 0 1 1 1 0 1 0] and **Ab** = [1 0 1 0 1 0 1 0 1 0]. To perform a match between these strings means to compare them so as to evaluate their affinity, as illustrated in Figure 6.10. If affinity is being measured via the complementarity between the two strings, then their affinity is equal to their Hamming distance. Else, if similarity corresponds to affinity, then their affinity is given by $L - D(\mathbf{Ag},\mathbf{Ab})$, where $D(\mathbf{Ag},\mathbf{Ab})$ is the Hamming distance between **Ag** and **Ab**.

Ag = [1 0 0 0 1 1 1 0 1 0]
Ab = [1 0 1 0 1 0 1 0 1 0]

Match(**Ab**,**Ag**): 0 0 1 0 0 1 0 0 0 0

Complementarity: $D(\mathbf{Ab},\mathbf{Ag}) = \sum$ Match (Affinity = 2)
Similarity: $L - D(\mathbf{Ab},\mathbf{Ag})$ (Affinity = 8)

Figure 6.10: Match between an antigen **Ag** and an antibody **Ab**, and their affinity assuming shape complementarity or similarity.

Two other affinity measures are broadly used in Hamming shape-spaces, namely, the so-called *r*-contiguous bits (*rcb*) and the *r*-chunks matching rules. In the *rcb* case, the number of contiguous matching bits (assuming the similarity between two strings is desired) determines the affinity between two strings. The match between two strings is performed and the affinity corresponds to the size of the longer sequence of matching (equal) bits. In *r*-chunks, only *r* contiguous positions are specified instead of specifying all *L* attributes of a string. An *r*-chunk detector can be thought of as a string of *r* bits together with its starting position within the string, known as its window (Esponda et al., 2004). Figure 6.11 illustrates the *r*-contiguous bit and the *r*-chunks affinity measures.

In Figure 6.11(a), if $r \leq 4$, then the antigen is recognized by the antibody; otherwise no recognition takes place. The affinity is the number of contiguous bits, thus, 4. In Figure 6.11(b) an antigen is represented by a string of length $L = 5$ and three chunks of length $r = 3$ are presented. Each of these chunks could correspond, for instance, to portions of antibodies. Thus, for $r = 3$ any antibody containing one of the chunks illustrated would recognize the antigen.

In Symbolic shape-spaces, the attribute strings that represent the components of the artificial immune system are composed of at least one symbolic attribute. Figure 6.12 illustrates a Symbolic shape-space containing two antibodies that represent a database of trips.

One last concept has to be discussed before proceeding with the main immune algorithms, for the determination of the affinity between two attribute strings is not enough to indicate if the two molecules bind. For instance, one may want to qualify a recognition and binding between two molecules only in cases their affinity is greater than or equal to a given threshold ε, termed *affinity threshold*.

Ag = [1 0 0 0 1 1 1 0 1 0]
Ab = [1 0 1 0 1 0 1 0 1 0]

Match(**Ab**,**Ag**): 1 1 0 1 1 0 1 1 1 1

(a)

Ag = [1 0 1 1 0]
d[1] = [1 0 1]
d[2] = [0 1 1]
d[3] = [1 1 0]

(b)

Figure 6.11: (a) The *r*-contiguous bit matching rule. (b) The *r*-chunks matching rule.

	Description	Date	Flight	Country	From	To	Price (£)
Antibody (Ab_1):	Business	1996	212	Brazil	Campinas	Greece	546.78
Antibody (Ab_2):	Holiday	2000	312	U.K.	London	Paris	102.35
Antigen (Ag):	Holiday	2000	212	U.K.	London	Greece	546.78
Match Ag – Ab_1:	0	0	1	0	0	1	1
Match Ag – Ab_2:	1	1	0	1	1	0	0

Figure 6.12: Example of a Symbolic shape-space representation in which some attributes are symbolic and others are numeric. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

The affinity threshold is important because it allows artificial immune systems to incorporate the concept of cross-reactivity, and thus it is also called *cross-reactivity threshold*. To illustrate the role of the cross-reactivity threshold when the Hamming distance, $D = \sum\text{Match}$, is used to evaluate the affinity between the bitstrings proportionally to their degree of similarity $L - D$ (L is the bitstring length), consider the example of Figure 6.13. In this example, the affinity between **Ab** and $\mathbf{Ag}_1$ is $aff(\mathbf{Ab},\mathbf{Ag}_1) = 8$, and the affinity between **Ab** and $\mathbf{Ag}_2$ is $aff(\mathbf{Ab},\mathbf{Ag}_2) = 6$. If an affinity threshold $\varepsilon = 8$ is used, then the antibody **Ab** recognizes $\mathbf{Ag}_1$, but does not recognize $\mathbf{Ag}_2$.

$\mathbf{Ab} = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0]$
$\mathbf{Ag}_1 = [1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$
$\mathbf{Ag}_2 = [0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0]$

Match($\mathbf{Ab}$,$\mathbf{Ag}_1$): 0 0 1 0 0 1 0 0 0 0
Match($\mathbf{Ab}$,$\mathbf{Ag}_2$): 1 1 0 1 0 0 0 1 0 0

Affinity $\mathbf{Ab}$-$\mathbf{Ag}_1$: 8
Affinity $\mathbf{Ab}$-$\mathbf{Ag}_2$: 6

Figure 6.13: If the cross-reactivity threshold is $\varepsilon = 8$, then $\mathbf{Ag}_1$ was recognized by **Ab**, while $\mathbf{Ag}_2$ was not.

6.3.3. Immune Algorithms

The literature is rich with works using particular aspects and principles of the immune system to design new algorithms or improve existing techniques for problem solving. However, given a suitable representation for the immune cells and molecules, and how to evaluate their interactions, it is possible to identify some general-purpose immune algorithms. These algorithms can be separated into two classes: population-based and network-based. The first class involves all algorithms that do not take into account the immune network, and the netwo-

rk-based algorithms are all those inspired by the network theory of the immune system. Five main classes of algorithms will be reviewed here:

- *Bone marrow*: used to generate populations of immune cells and molecules to be used in AIS (and possibly in other approaches, such as genetic algorithms).
- *Negative selection*: used to define a set of detectors (e.g., attribute strings) to perform, mainly, anomaly detection.
- *Clonal selection*: used to generate repertoires of immune cells driven by antigens. It regulates the expansion, genetic variation, and selection of attribute strings.
- *Continuous immune network models*: used to simulate dynamic immune networks in continuous environments.
- *Discrete immune network models*: used to simulate dynamic immune networks in discrete environments.

6.4 BONE MARROW MODELS

The bone marrow is the site where all blood cells, including the immune cells, are generated. The simplest bone marrow model simply generates attribute strings of fixed length L using a (pseudo) random number generator. For real-valued shape-spaces, the interval in which a molecule m is going to be defined, e.g., $m \in [0,1]^L$, has to be chosen. In Hamming shape-spaces, the string that represents m is randomly generated from elements belonging to a predefined alphabet, e.g., $m \in \{0,1\}^L$ for binary strings (bitstrings). In the case of Integer shape-spaces, an algorithm to perform a random permutation of L elements can be used.

Some more complex and biologically plausible bone marrow models use gene libraries from which the immune cells and molecules are rearranged or evolved, as illustrated in Figure 6.14. In this picture, two libraries of genes are available and one gene from each library is chosen and concatenated with the others so as to form an antibody molecule. Assuming, for instance, that each gene has 6 bits, the final **Ab** molecule has a total length $L = 12$ bits.

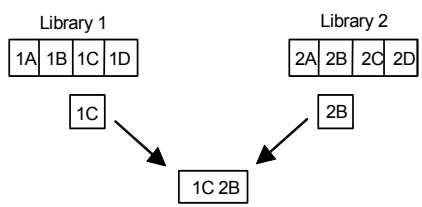


Figure 6.14: Generation of an antibody molecule by concatenating genes from different gene libraries. (Compare this figure with Figure 6.3.)

```
procedure [Ab] = gene_library(N)
    initialize L     //initialize all libraries
    for i from 1 to N do,          //N Abs will be generated
        for j from 1 to n do,      //for each library
             gene_index ← rand(L_l) //select a gene randomly
             Ab_ij ← L(j,gene_index)//each gene of Ab_i
        end for
    end for
end procedure
```

Algorithm 6.1: Procedure to generate antibodies from gene libraries.

One particular feature of bone marrow models based upon gene libraries is that not all genes present in the *genotype* (total collection of genes) are expressed in the *phenotype* (expressed antibody molecules). Thus, with a small finite genome it is possible to generate a very large number of different molecules. Besides, the use of specific genes in the libraries allows us to store previous and useful information in the gene segments, thus naturally introducing previous knowledge in the system. To illustrate this, take the idea of using an AIS to solve the traveling salesman problem. In this case, assume that you know some good partial routes in the TSP that you want to maintain or privilege in your system. These partial routes can be encoded in some gene segments to be used in the generation of initial candidate solutions to the AIS, or any other search strategy, including evolutionary algorithms.

Let **L** be the set of n libraries initialized either at random or using any predefined knowledge. The number of libraries is arbitrary and is usually defined by the problem at hand. As one gene is taken from each library and each gene has a length L_g, the number of libraries will be a function of the length desired for the molecules. For instance, if a molecule of length 24 is desired and each gene has a length $L_g = 8$, then three libraries are required. Algorithm 6.1 summarizes the procedure used to construct antibody molecules from gene libraries of length L_l assuming a number N of antibodies to be generated. Note that N antibodies are generated at random by concatenating one gene from each library; each gene has a length $L_g \geq 1$. The procedure returns a matrix of antibodies **Ab** of dimension $N \times (g.l)$, where g is the length of the genes and l is the number of libraries (assuming one gene is taken from each library).

6.4.1. Selected Applications from the Literature: A Brief Description

Although we suggested that more complex bone marrow models, such as the one that uses gene libraries to construct receptor molecules, can be used as means to naturally account for previous knowledge in artificial immune systems, not much has been done in this direction. Hart and Ross (1999) and Coello Coello et al. (2003) used gene libraries for scheduling problems, and Kim and Bentley (1999a) used concepts from gene libraries to generate detectors for network intrusion.

Most AIS use simple algorithms to generate the initial repertoires of molecules, and models based on gene libraries have basically been used to study the biological immune system and its theoretical models. So, this section briefly reviews two applications of gene libraries in contexts that are more related to biological modeling than to problem solving. The example of an application of gene libraries for solving scheduling problems is also briefly discussed. Some exercises in the end of the chapter explore the use of bone marrow models to generate candidate solutions for other problems.

Evolution of the Genetic Encoding of Antibodies

Hightower et al. (1995) and Perelson et al. (1996) used a genetic algorithm and a binary Hamming shape-space to study the effects of evolution on the genetic encoding for antibody molecules in the immune system.

The genetic material for one antibody molecule was stored in five separate gene libraries. Producing one antibody molecule begins with a random selection of a gene segment from each of the libraries. In their model, the bitstring representing the genotype of an individual was divided into four equal-size libraries of antibody segments (see Figure 6.14). Within each library there were eight elements, represented as bitstrings of length sixteen, so each individual genome had a total of 512 bits. The expressed antibodies had a length of 64 bits.

The experiments showed that the GA could optimize complex genetic information, being able to organize the structure of the antibody libraries. They also showed that the selection pressure operating on the phenotype as a whole could translate to selection pressure acting on individual genes, even though not all genes were expressed in the phenotype.

Antigenic Coverage and Evolution of Antibody Gene Libraries

Oprea and Forrest (1998) were intrigued by how the vertebrate immune system is capable of binding with any type of antigen, even if this antigen has never been encountered before. In order to better understand how this coverage is achieved, they used an evolutionary algorithm to explore the strategies that the antibody libraries may evolve in order to encode antigenic sets of various sizes. They derived a lower and an upper bound on the performance of the evolved antibody libraries as a function of their size and the length of the antigen string. They also provided some insights in the strategy of antibody libraries, and discussed the implications of their results for the biological evolution of antibody libraries.

Their work was based on that of Hightower et al. (1995) and also employed bitstrings to represent antibodies and antigens. They tested several different antibody repertoires with the same number of antibodies. The affinity between any two strings was proportional to the degree of similarity between two strings - length minus the Hamming distance - and each antigen is presented to all antibodies in the system. The fitness of an antibody is given by the average affinity over the whole set of antigens.

In their discussion, they suggested that (somatic) mutation allows the affinity maturation of antibodies that recognize antigens. Thus, there is an evolutionary trade-off between increasing the size of the antibody libraries and improving the efficiency of the somatic mutation process. From the perspective of AIS, this result is important for it suggests that affinity maturation can compensate for an inappropriate choice (or evolution) of antibody libraries.

Generating Antibodies for Job Shop Scheduling

Scheduling is a terminology used to refer to a class of planning problems in which a specific set of tasks has to be executed in a certain period of time subjected to limited resources and other types of constraints. In brief, scheduling is the problem of allocating resources through time (Rardin, 1998). *Job shop scheduling* (JSS) is a scheduling problem in which there is a production plant with different types of machines and a certain number of jobs to be completed, each one corresponding to a specific task. Solving a job shop scheduling problem corresponds to determining what tasks from the different jobs have to be executed in what order so as to minimize the production cost.

Coello Coello et al. (2003) used an artificial immune system based on gene libraries and clonal selection for solving the job shop scheduling problem. To use gene libraries to construct antibody molecules (candidate solutions) for solving the JSS problem, the authors employed a special representation. Each element of the library corresponds to the sequence of jobs processed by each of the machines. An antibody is thus an attribute string with the job sequence processed by each of the machines.

Their procedure works as follows. An antibody library (i.e., set of strings that encode different job sequences for each machine) is initially generated at random. An antigen, which is a candidate solution to the JSS problem (i.e., a sequence of jobs) is also generated randomly. Then, a single antibody is created by combining different segments from the library, in a way similar to that illustrated in Figure 6.14. This antibody is decoded such that one can read the job sequence for each machine and a local search algorithm is used to improve this potential solution. The solution encoded by the antibody is then compared with that of the antigen, and in case it is better it replaces the antigen. Note that the antigen is a sort of memory of the best solution found during the iterative process of adaptation. A clonal selection procedure (Section 6.6) is the applied and the best segments produced are used to update the antibody library.

6.5 NEGATIVE SELECTION ALGORITHMS

One of the main functions of the thymus is to promote the maturation of T-cells. Immature T-cells, generated in the bone marrow, migrate into the thymus where some of them differentiate into immunocompetent cells (cells capable of acting during an adaptive immune response), and others are purged from the repertoire due to a strong recognition of self. This process of eliminating cells whose receptors recognize self is known as *negative selection*. The thymic negative se-

lection has to guarantee that the T-cell repertoire that leaves the thymus and goes to the periphery does not contain cells that recognize self cells and molecules. Thus, the immature T-cells that maturate and leave the thymus become a sort of change or anomaly detectors.

The thymic negative selection has inspired the development of a negative selection algorithm with applications focusing, mainly, anomaly detection. There are two main versions of this algorithm: one for binary shape-spaces and another for real-valued shape spaces.

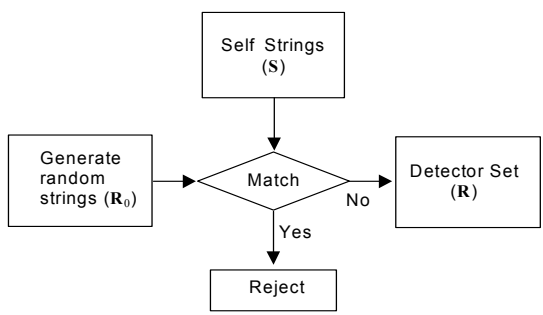
6.5.1. Binary Negative Selection Algorithm

Forrest et al. (1994) proposed a change detection algorithm inspired by the negative selection of T-cells within the thymus. This procedure was named *negative selection algorithm,* and its original application was in computational security, though many other applications were proposed. A single type of immune cell was modeled: T-cells represented as bitstrings of length L.

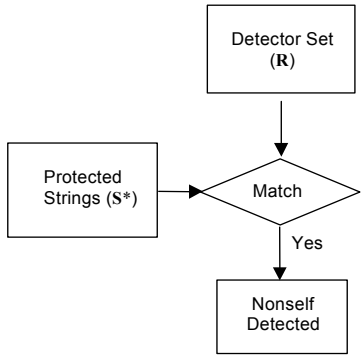
The algorithm is divided into two phases: a *censoring phase*, and a *monitoring phase*. In the censoring phase, given a known set of self patterns, named self-set **S**, the T-cell receptors will have to be tested for their capability of recognizing and binding with the self patterns. If the receptor on a T-cell (attribute string) recognizes a string from the self set, it is discarded; else it is selected as an immunocompetent cell and enters a set of *detectors*. The detectors are then used to monitor the system for anomalies, i.e., unusual patterns or behaviors.

The censoring phase of the negative selection algorithm is depicted in Figure 6.15(a) and can be summarized as follows:

1. *Initialization*: assuming a set **S** of self patterns (strings) to be protected, randomly generate attribute strings and place them in a set $\mathbf{R}_0$.
2. *Affinity evaluation*: test if the strings that are being generated in $\mathbf{R}_0$ match any of the strings in the self set **S** by determining their affinity.
3. *Generation of the detector set*: if the affinity of a string from $\mathbf{R}_0$ with at least one string from **S** is greater than a given threshold, then this string recognizes a self pattern and has to be eliminated (negative selection); else the string is introduced into the detector set **R**.



(a)



(b)

Figure 6.15: Negative selection algorithm. (a) Generation of valid detector set (censoring). (b) Monitoring of protected data.

After generating a set of detectors, the algorithm can be used to monitor changes in the self set or to track novelties. In a *monitoring* phase (Figure 6.15(b)), a set **S*** of protected strings is matched against the elements of the detector set. The set **S*** may be the set **S** with no change, a completely new set, or can be composed of elements of **S** perturbed with noise or added with new elements. If a recognition occurs between a protected string and a string from the detector set, then a nonself pattern (string) is detected. Figure 6.15(a) illustrates the censoring phase of the negative selection algorithm as introduced by Forrest et al. (1994).

```
procedure [R] = NSA_censor(S,ε,N,L)
    t ← 1
    while t <= N do, //N detectors will be generated
        r0 ← rand(L) //randomly generate a string of length L
        flag ← 0
        for every s of S do, //for each element of S (self)
             aff ← affinity(r0,s) //determine affinity
             if aff >= ε then
                 flag ← 1; break
             end if
        end for
        if flag == 0,
            R ← add(R,r0)   //add r0 to the set of detectors R
            t ← t + 1
      end if
    end while
end procedure
```

Algorithm 6.2: Procedure to implement the censoring phase of the negative selection algorithm (NSA).

```
procedure [flag] = NSA_monitor(S*,R,ε)
    for every s of S* do,  //for all elements of S*
       for every r of R do,
           aff ← affinity(r,s)   //determine affinity
           if aff >= ε then
              flag ← nonself detected  //flag
           end if
       end for
    end for
end procedure
```

Algorithm 6.3: Procedure to implement the monitoring phase of the negative selection algorithm.

Let **S** be the self set, ε the affinity threshold (i.e., the minimum degree of match to qualify a recognition between two molecules), and N the total number of detectors of length L to be generated and placed in $\mathbf{R}_0$. The censoring phase of the negative selection algorithm that returns a set **R** of detectors can be implemented using the pseudocode presented in Algorithm 6.2.

After a set of detectors has been generated, the system can be monitored for anomalies. Algorithm 6.3 describes a procedure to monitor anomalies. It receives as inputs the set of detectors **R**, a set of protected strings **S***, and the affinity threshold. Its output can be the patterns from **S*** that were detected as a nonself (anomaly), or simply a message (flag) indicating that nonself patterns were detected.

6.5.2. Real-Valued Negative Selection Algorithm

González and Dasgupta (2003) introduced a new negative selection algorithm based on a real-valued shape-space. The *real-valued negative selection algorithm* (RNS) is rooted in the same ideas as the original binary negative selection algorithm with the main difference that the self and nonself spaces are subsets of $\Re^L$, particularly $[0,1]^L$.

A detector (antibody) in the RNS algorithm is a hyper-sphere with radius ε in an L-dimensional real-valued shape-space $\Re^L$. The proposed matching between real-valued attribute strings (i.e., detectors, elements from the self set, and monitored data) is expressed as a membership function of the detectors **r** in relation to the elements $\mathbf{s} \in \mathbf{S}$, $\mu_{\mathbf{r}}(\mathbf{s})$, and is directly proportional to their Euclidean distance $\|\cdot\|$ and inversely proportional to the radius of the detector (affinity threshold) ε, as follows

$$\mu_{\mathbf{r}}(\mathbf{s}) = \exp(-\|\mathbf{r}-\mathbf{s}\|^2/2\varepsilon^2) \qquad \textbf{(6.4)}$$

Similarly to the binary version of the algorithm, the input is a set **S** of self patterns, but represented by real-valued vectors instead of binary strings. A number N of detectors is initially generated at random and updated so as to cover the nonself space by moving detectors away from self vectors and keeping detectors separated from each other in order to maximize the coverage of the nonself

space. Note that the approach here is based on moving detectors about the space and varying their affinity threshold, instead of constantly generating random detectors and comparing them with the elements of the self set **S**.

In order to determine the match between a detector **r** and a self-point, the *k*-nearest neighbors from **S** in relation to **r** are determined; that is, a number *k* of points from **S** that present the smallest distance to **r** are selected. The median distance of these *k* neighbors is calculated and if this is less than the affinity threshold ε, then the detector is assumed to match self.

Each candidate detector that recognizes an element from the self set has an associated age that is increased at each iteration. When a candidate detector reaches a certain age *T* and has not been able to move away from the self set it is replaced by a new, randomly generated, candidate detector.

To ensure that the algorithm will converge to a stable state the authors proposed the use of a decaying adaptation rate $\eta(t)$ for the algorithm, where *t* is the time index

$$\eta(t) = \eta(0).\exp(-t/\tau) \tag{6.5}$$

where $\eta(0)$ is the initial value for $\eta(t)$ and τ is a parameter that controls its decay. The stopping criterion is a prespecified maximum number of iterations.

```
procedure [R] = RNS_censor(S,ε,N,L,T,η0,k,max_it)
    t ← 1
    R ← rand(N,L)   //generate N detectors of length L
    while t <= max_it do,

       for every r of R do, //for each detector
            NearCells ← k-nearest neighbors of r in S
            NearestSelf ← median of ordered NearCells
            if distance(r,NearestSelf) < ε then
                Δr = η(t)*(∑c∈NearCells(r−c))/|NearCells|
                if age of r >  T
                     Replace r by a new random detector
                else
                     Increase age of r
                     r ← r + Δr
                end if
            else
                age of r ← 0
                Δr = η(t)*(∑r′∈Rμr(r′)(r−r′))/∑r′∈Rμr(r′)
                r ← r + Δr
            end if
       end for
       t ← t + 1
    end while
end procedure
```

Algorithm 6.4: Procedure to implement the censoring phase of the real-valued negative selection algorithm (RNS).

Let **S** be the self set to be protected, ε the affinity threshold, N the number of detectors of length L, T the maturity age of the detectors, η_0 the initial value for the updating rate, k the number of neighbors for defining a self-match, and `max_it` the maximum number of iterations allowed. Algorithm 6.4 summarizes the real-valued negative selection algorithm as proposed by González and Dasgupta (2003). The equations used to move the detectors in the space are embodied in the procedure. After the detectors are generated, an anomaly can be detected by matching the monitored patterns, **S***, against the detectors. If an element from **S*** lies within the hypersphere of a detector; that is, if its distance to a detector is less than ε, then it is recognized as a nonself. Otherwise, it is a self.

6.5.3. Selected Applications from the Literature: A Brief Description

Negative selection algorithms are among the most well-known and applied immune algorithms. As they are usually used to generate a set of detectors to identify anomalies, their application areas have emphasized computer security against viruses and network intrusion detection. Besides, the application of artificial immune systems to computer security is very intuitive: "if we have an immune system capable of protecting us against viral infections, why don't we extract some of these ideas to develop an artificial immune system for computers?" Negative selection is the main algorithm used in this endeavor. This section reviews two applications of the negative selection algorithms to the problem of detecting anomaly. The first example describes the application of the binary negative selection algorithm to computer network traffic anomaly detection. The second example involves the use of RNS for cancer diagnosis.

Network Intrusion Detection

Hofmeyr and Forrest (2000) focused on the problem of protecting a local area broadcast network of computers (LAN) from network-based intrusions. The negative selection algorithm was the main immune principle employed in this application and will be the focus of this section. However, several other concepts were also incorporated such as co-stimulation, presentation of processed antigens by phagocytes, and the use of both B-cells and T-cells.

Broadcast LANs have the property that every computer in the network has access to each packet passing through the LAN. The aim of their artificial immune system for network security is to monitor the network traffic through the network connections. A connection is defined in terms of its *data-path triple*: the source internet protocol (IP) address, the destination IP address, and the service (port) by which the computers communicate (Figure 6.16). Each data-path triple corresponds to a network connection and is represented using a binary string of length $L = 49$ in a binary Hamming shape-space. Self is defined as the set of normally occurring connections observed over time on the LAN. By contrast, nonself consists of those connections not normally observed on the LAN, and are assumed not known *a priori*. In this case, the AIS is required to learn to discriminate between what is self (normally occurring connections) and what is nonself (intrusions).

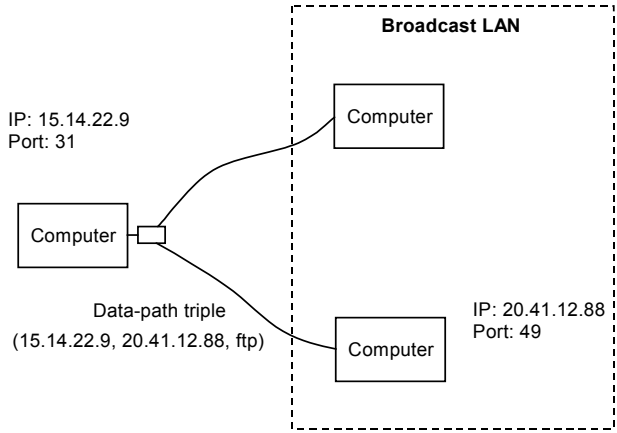


Figure 6.16: Example of a broadcast LAN and its connections. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

The detector set is generated based on the binary version of the negative selection algorithm as follows. Each binary string of length $L = 49$ representing a detector is created randomly and remains immature for a time period t, known as the tolerization period. During this period, the detector is exposed to the self strings, and if a recognition occurs, then the detector is eliminated. If it does not recognize any string during tolerization, it becomes a mature detector. Mature detectors need to exceed a cross-reactivity threshold (r-contiguous bits) in order to become activated; this increases its life span. Recognition is quantified by matching two strings using the r-contiguous bit rule.

These nonself detectors are constantly being matched against the bitstrings representing the data-path triples that correspond to the connections in the network. Each detector is allowed to accumulate matches through a *match count* (m), which itself decays over time. In order for a detector to become activated, it has to match at least δ strings within a given time period; this threshold δ was termed the *activation threshold*. Once a detector has been activated, its match count is reset to zero.

When a new packet (bitstring) enters the network, several different detectors are activated with distinct degrees of affinity (through the r-contiguous bit rule). The best matching detectors become memory detectors (memory cells). These memory detectors make copies of themselves that are then spread out to neighboring nodes in the network. Consequently, a representation of the memory detector is distributed throughout the network, accelerating future (secondary) responses to this specific nonself bitstring.

When a detector is activated, it sends an initial signal that an anomaly has been detected. As in the biological immune system, the detection of an anomaly is not enough to promote an immune response; a co-stimulatory signal is required from an accessory cell. The authors simulated the co-stimulatory (or

second) signal by using a human operator, who is given a time period t_s to decide if the detected anomaly is really nonself. If the operator decides that this anomaly is indeed nonself, then a second signal is returned to the detector that identified it. This second signal increases the life span of this detector, maintaining it as a memory detector. The recognition of an anomaly in the absence of the second signal causes the death of a detector and its replacement by a new one. Figure 6.17 summarizes the life cycle of a detector.

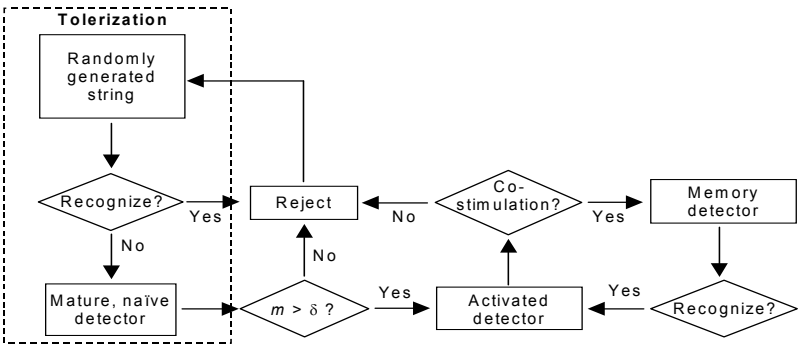


Figure 6.17: Life cycle of a detector. (Note that the negative selection algorithm is embodied in the tolerization period.) (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

Breast Cancer Diagnosis

Among the many problems González and Dasgupta (2003) used to assess the performance of the real-valued negative selection algorithm is the breast cancer diagnosis dataset (Wolberg and Mangasarian, 1990), available at (Newman et al., 1998). This dataset was obtained by Dr. William H. Wolberg from the University of Wisconsin Hospitals and contains 699 samples. Each sample corresponds to different variations of 10 cell features (e.g., cell size, shape, etc.) and, based on these features, an assigned class label (benign or malignant). Thus, the data set is composed of 699 input patterns with 10 attributes each plus a class label. Among the 699 patterns, 16 had missing attributes and these were not used in the experiments. The data set was divided into two parts: one for training, with 271 benign samples; and one for testing, with 412 mixed benign and malignant samples.

In order to perform anomaly detection, the authors used a hybrid approach combining the real-valued negative selection algorithm with a supervised classification algorithm. In particular, they used a multi-layer feedforward network trained with the standard backpropagation algorithm (Section 4.4.4) as a classifier. The hybrid system is as follows. Using only the 271 benign samples, the RNS algorithm was employed to generate samples of anomalies (malignant samples), which altogether (sampled benign + generated malignant) served to

train the MLP network. The MLP network was then responsible for classifying the new input data (**S***) into self (benign) or nonself (malignant).

When the goal is to assess the effectiveness of the anomaly detection process, two measures commonly used are the *detection rate* (*DR*) and the *false alarm rate* (*FAR*), which can be defined as follows:

$$DR = TP/(TP+FN) \quad \textbf{(6.6)}$$

$$FAR = FP/(FP+TN) \quad \textbf{(6.7)}$$

where *TP* is the number of *true positives* (anomaly identified as anomaly), *FP* is the number of *false positives* (normal identified as anomaly), *TN* is the number of *true negatives* (normal identified as normal), and *FN* is the number of *false negatives* (anomaly identified as normal).

400 samples of the malignant class were generated using RNS with $\varepsilon = 0.1$, $\eta_0 = 1$, $T = 5$ and $k = 1$. The RNS algorithm was allowed to run for 400 iterations, `max_it` = 400. A trade-off between the detection rate (*DR*) and the false alarm rate (*FAR*) was used to evaluate the performance of the algorithm.

6.6 CLONAL SELECTION AND AFFINITY MATURATION

The clonal selection principle is used to describe the basic features of an adaptive immune response to antigens. It establishes the idea that only those cells that recognize the antigens proliferate, thus being selected against those that do not. The selected cells are subject to an affinity maturation process, which improves their affinity to the selective antigens. Some features are important for the development of a clonal selection algorithm (Forrest et al., 1993; de Castro and Timmis, 2002):

- Antigens are typically encountered sequentially, and the immune system responds with a subset of its immune cells - those that come into contact with the antigen.
- An antigen selects several immune cells to proliferate (cloning). The proliferation rate of each immune cell is proportional to its affinity with the selective antigen: the higher the affinity, the higher the number of offspring generated; and vice-versa.
- The mutation suffered by each immune cell during reproduction is inversely proportional to the affinity of the cell receptor with the antigen: the higher the affinity, the smaller the mutation rate, and vice-versa.

Forrest and collaborators (Forrest et al., 1993) focused on the first feature listed - the locality of the antigenic presentation - and claimed that a genetic algorithm without crossover is a reasonable model of clonal selection, while the GA with crossover models genetic evolution. With an independent view, de Castro and Von Zuben (2002) focused on the affinity proportional reproduction and mutation of clonal selection and suggested that clonal selection is more than a GA without crossover, though it can be characterized as an evolutionary algorithm. Both algorithms can be applied to pattern recognition and will be revi-

ewed separately. Other Clonal selection algorithms have been proposed in the artificial immune systems literature, such as CLIGA (Cutello and Nicosia, 2004) and the B-cell algorithm (Kelsey and Timmis, 2003), but these are not discussed here.

6.6.1. Forrest's Algorithm

Forrest et al. (1993) employed a binary Hamming shape-space to model clonal selection and affinity maturation. The goal was to study the pattern recognition and learning that takes place at the individual and species level of this artificial immune system. In their model, a population of antibodies was evolved in order to recognize a given population of antigens. A genetic algorithm was used to study the maintenance of diversity and generalization capability of their AIS, where generalization means the detection of common patterns shared among many antigens. The affinity function between an antibody and an antigen was evaluated by summing the number of complementary bits between their attribute strings (i.e., Hamming distance). This way, it was possible to study the ability of a GA in the detection of common patterns in the antigen population, and to discern and maintain a diverse population of antibodies.

As a result of their study about diversity in a model of the immune system, the authors proposed an immune algorithm that has been used in important applications of AIS, such as computer network security (Kim and Bentley, 1999b) and job-shop scheduling (Hart and Ross, 1999). The authors proposed the following procedure:

1. Build random populations of antigens and antibodies.
2. Match antibodies against antigens and determine their affinity.
3. Score the antibodies according to their affinity.
4. Evolve the antibody repertoire using a standard GA.

As can be observed from the procedure described above, the authors proposed to evolve a population of antibodies towards the antigens using a standard genetic algorithm, that is, a GA with crossover. However, as suggested by themselves, a GA without crossover is a reasonable model of clonal selection. Thus, they also performed some experiments with a GA without crossover, and claimed that similar performances were obtained when compared with a GA with crossover. However, to evolve a population of antibodies that maintains diversity under the GA, a fitness measure for the antibodies was proposed as follows:

1. Choose an antigen randomly.
2. Choose, without replacement, a random sample of μ antibodies.
3. Match each antibody against the selected antigen.
4. Add the match score of the highest affinity antibody to its fitness maintaining the fitness of all other antibodies fixed.
5. Repeat this process for many antigens (typically three times the number of antibodies).

The authors argued that this model allows the study of how the immune system learns which antibodies are useful for recognizing the given antigens. In

addition, it was suggested that change of details such as the shape-space used and the sampling methods for the antibody molecules could create several interesting variations of this algorithm.

Let **S** be the set of N antigens to be recognized, `max_it` the maximum number of iterations for the GA, *pc* the crossover probability (if crossover is used), *pm* the mutation probability, and **P** the antibody repertoire. Algorithm 6.5 summarizes Forrest's clonal selection algorithm.

```
procedure [M] = CSA(S,N,max_it,pc,pm)
    initialize P
    gl ← 3*N              //three times the number of Abs
    t ← 1
    while t <= max_it do,//for g iterations
      for j from 1 to gl do,
           s_j ← select_random(S) //select an Ag randomly
           P_μ ← select(P,μ)      //select μ Abs from P
           //best matching antibody w with match score m
           [w,m] ← affinity(P_μ,s_j)
           fit_w ← fitness(w,m)  //update the winner fitness
      end for
      P ← select(P,fit)
      P ← reproduce(P,fit,pc)
      P ← variate(P,fit,pm)
      j ← j + 1
  end while
end procedure
```

Algorithm 6.5: Procedure to implement Forrest's clonal selection algorithm. (Note that the last steps are those of an evolutionary algorithm.)

6.6.2. CLONALG

The clonal selection algorithm, named CLONALG, proposed by de Castro and Von Zuben (2002) takes into account the other two important features of clonal selection not accounted for by Forrest and collaborators; namely, affinity proportional selection and mutation. On the other side, CLONALG does not consider the locality of an immune response. CLONALG was initially proposed to perform pattern recognition and then adapted to solve multimodal function optimization tasks, as will be illustrated in the selected applications. Given a set of patterns to be recognized (**S**), the basic steps of the CLONALG algorithm are as follows (de Castro and Timmis, 2002):

1. *Initialization*: create an initial population of individuals (**P**).
2. *Antigenic presentation*: for each pattern from **S**, do:
   2.1. *Affinity evaluation*: present it to the population **P** and determine its affinity with each element of the population **P**.

2.2. *Clonal selection and expansion*: select n_1 highest affinity elements of **P** and generate clones of these individuals proportionally to their affinity with the antigen: the higher the affinity, the higher the number of copies, and vice-versa.

2.3. *Affinity maturation*: mutate all these copies with a rate inversely proportional to their affinity with the input pattern: the higher the affinity, the smaller the mutation rate, and vice-versa. Add these mutated individuals to the population **P** and reselect the best individual to be kept as the memory **m** of the antigen presented.

2.4. *Metadynamics*: replace a number n_2 of low affinity individuals by (randomly generated) new ones.

3. *Cycle*: repeat Step 2 until a certain stopping criterion is met.

```
procedure [M] = CLONALG_PR(S,max_it,n1,n2)
    initialize P                    //P ⊇ M
    t ← 1
    while t <= max_it do,  //for max_it iterations
       for every s of S do, //for all patterns (antigens)
            aff ← affinity(s,P)  //determine the affinity
            P1 ← select(P,n1,aff)//n1 highest affinity
            C ← clone(P1,aff)    //proportional to affinity
            C1 ← mutate(C,aff)   //inversely proportional
            aff ← affinity(s,C1) //affinity of clones with s
            //highest affinity is candidate to become memory
            m ← select(C1,aff)
            //m replaces the current memory cell for s if it
            //has a greater affinity
            M ← insert(M,m)
            //replace n2 low affinity cells
            P ← replace(P,n2)
       end for
       t ← t + 1
    end while
end procedure
```

Algorithm 6.6: The clonal selection algorithm, CLONALG, to pattern recognition. (There is one memory cell for each pattern to be recognized.)

Matrix $\mathbf{M} \subseteq \mathbf{P}$ with the difference that the elements of **M** are only replaced by new elements of higher affinity. This process, together with the affinity proportional mutation, promotes a greedy search of the affinity landscape.

Let **S** be the set of patterns (antigens) to be recognized, `max_it` the number of iterations to be performed, N the size of the population, n_1 the number of high affinity antibodies to be selected for cloning, and n_2 the number of low affinity antibodies to be replaced at each iteration ($n_1 + n_2 \leq N$). Algorithm 6.6 contains the pseudocode for implementing the clonal selection algorithm CLONALG to perform pattern recognition. The number of memory antibodies in this case is

the same as the number of patterns to be recognized. The algorithm assumes one memory cell for each antigen, and a memory cell is only replaced by memory cells with higher affinity with a given antigen. It returns as output the set of memory cells **M**. The immune repertoire **P** is composed of the memory cells **M** plus a number of remaining cells **R**.

To apply the proposed CLONALG algorithm to optimization tasks, a few modifications have to be made in the algorithm as follows (de Castro and Von Zuben, 2002). **P** corresponds to a repertoire of antibodies (chromosomes in evolutionary algorithms), and $g(\cdot)$ is the objective function used to evaluate the quality (fitness) of each antibody.

- In Step 2, there is no explicit antigen population to be recognized, but an objective function $g(\cdot)$ to be optimized (maximized or minimized). This way, an antibody affinity corresponds to the evaluation of the objective function for the given antibody, i.e., it corresponds to the *fitness* of the antibody.
- As there is no specific antigen population to be recognized, the whole antibody population will compose the memory set and, hence, it is no longer necessary to maintain a separate memory set.
- Instead of selecting the single best antibody for each antigen, n antibodies are selected to compose the antibody set.

The CLONALG algorithm to solve optimization tasks can be described as follows:

1. *Initialization*: create an initial population of antibodies (**P**).
2. *Fitness evaluation*: determine the fitness of each element of **P**.
3. *Clonal selection and expansion*: select n_1 highest fitness elements of **P** and generate clones of these antibodies proportionally to their fitness: the higher the fitness, the higher the number of copies, and vice-versa.
4. *Affinity maturation*: mutate all these copies with a rate that is inversely proportional to their fitness: the higher the fitness, the smaller the mutation rate, and vice-versa. Add these mutated individuals to the population **P**.
5. *Metadynamics*: replace a number n_2 of low fitness individuals by (randomly generated) new ones.
6. *Cycle*: repeat Steps 2 to 5 until a certain stopping criterion is met.

If the optimization process aims at locating multiple optima within a single population of antibodies, then two parameters may assume default values. Assign $n_1 = N$, i.e., all antibodies from the population will be selected for reproduction; and the affinity proportionate cloning is not necessarily applicable, meaning that the number of clones generated for each of the N antibodies is assumed to be the same. This implies that each antibody will be viewed locally and have the same clone size (number of offspring) as the other ones, not privileging anyone for its affinity (fitness). The fitness will only be accounted for to determine the mutation rate of each antibody, which is still inversely proportional to the fitness.

```
procedure [P] = CLONALG_OPT(max_it,n1,n2)
    initialize P                    //P ⊇ P1
    t ← 1
    while t <= max_it do,           //for max_it iterations
        f ← eval(P)           //determine the fitness
        P1 ← select(P,n1,f)//select the n1 highest fitness
        C ← clone(P1,f)       //proportional to affinity
        C1 ← mutate(C,f)      //inversely proportional
        f1 ← eval(C1)         //fitness of the clones
        P1 ← select(C1,n1,f1)     //select the n1 best clones
        P ← replace(P,n2)     //replace n2 low fitness cells
        t ← t + 1
    end while
end procedure
```

Algorithm 6.7: The clonal selection algorithm, CLONALG, to optimization.

Following the same notation as before, Algorithm 6.7 summarizes the optimization version of the CLONALG algorithm. In this case, the subpopulation **P** could be viewed as the set of memory cells, but this is not explicitly accounted for in the algorithm.

6.6.3. Selected Applications from the Literature: A Brief Description

In order to illustrate the application of the CLONALG algorithm to perform learning and memory acquisition, it is going to be applied to a pattern recognition problem; the same problem used to illustrate the application of various artificial neural networks in Chapter 5. CLONALG is also going to be applied to a multimodal optimization task in order to evaluate its capability of locating and maintaining a stable population of multiple optima solutions.

Pattern Recognition

The goal of this example is to demonstrate that clonal selection together with affinity maturation can produce individuals with increasing affinities. In other words, this example shows that a blind search procedure is capable of extracting information from an unknown environment. Assume that the antigen population to be learned is represented by the benchmark set of eight binary characters we have been using in this book, as illustrated in Figure 6.18. Each character is represented by a bitstring of length $L = 120$ (the resolution of each picture is 12×10). The antibody repertoire is composed of $N = 10$ individuals, where $m = 8$ of them belong to the memory set **M**. The other running parameters were `max_it` = 500, $n_1 = 5$, $n_2 = 0$ and $\beta = 10$, where β is a parameter that defines the number of clones to be generated for each antibody, as given by Equation (6.8):

$$Nc = (\beta.N)/i \tag{6.8}$$

Figure 6.18: Set of antigens (input patterns) to be used to evaluate the CLONALG potential for pattern recognition.

where N is the total number of antibodies, and i is the index that determines the ranking of an antibody in relation to the others. Thus, the best matching (highest affinity) antibody from the repertoire is going to generate $\beta.N$ offspring, the second best matching antibody generates $(\beta.N)/2$, the third best antibody generates $(\beta.N)/3$, and so on. Equation (6.8) quantifies the affinity proportional reproduction of clonal selection.

Figure 6.19 illustrates the behavior of the pattern recognition version of CLONALG when applied to the problem depicted in Figure 6.18. Figure 6.19(a) depicts the initial memory set, and Figure 6.19(b) to (d) represents the maturation of the memory set (immune response) with time.

The affinity measure takes into account the length L of the strings minus the Hamming distance (D) between an antigen **Ag** and an antibody **Ab**, as illustrated in Figure 6.10. In this case, the smaller the distance, the lower the affinity, and vice-versa.

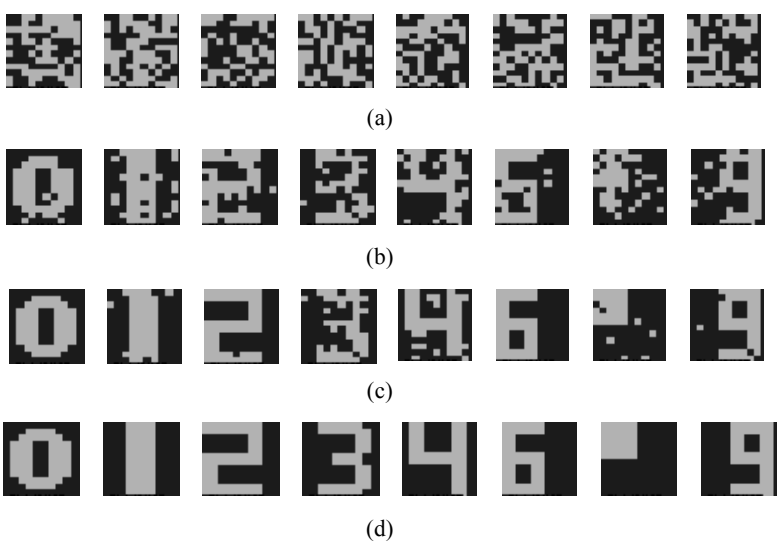


Figure 6.19: CLONALG applied to the pattern recognition task presented in Figure 6.18. (a) 0 iterations. (b) 100 iterations. (c) 200 iterations. (d) 500 iterations.

Multimodal Function Optimization

Consider the case of optimizing the function $g(x) = 2^{-2((x-0.1)/0.9)^2}\left(\sin(5\pi x)\right)^6$, studied in Chapter 3 and Chapter 4. Each antibody was represented using a binary string, with the same encoding scheme as the one adopted in Section 3.5.3. The running parameters adopted were `max_it` = 50, $n_1 = N = 50$, $n_2 = 0$, and $\beta = 0.1$, where $Nc = \beta.N$ defines the number of clones Nc to be generated for each antibody. For the parameters chosen, each antibody generates $Nc = \beta.N = 0.1 \times 50 = 5$ offspring. Figure 6.20 presents one simulation result obtained by CLONALG. Note that all the peaks of the function could be determined by the algorithm.

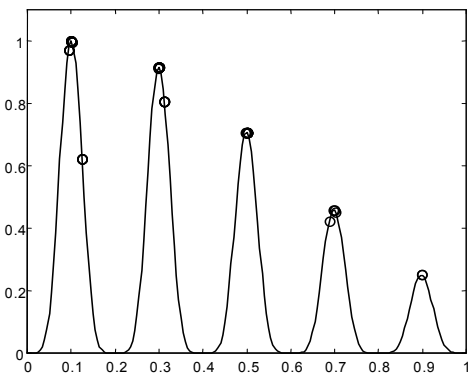


Figure 6.20: One simulation result of the optimization version of CLONALG when applied to function $g(x) = 2^{-2((x-0.1)/0.9)^2}\left(\sin(5\pi x)\right)^6$.

6.7 ARTIFICIAL IMMUNE NETWORKS

Following the network proposal of Jerne (1974), several network models were introduced by theoretical researchers interested in studying and modeling the immune system. The immune network is different from clonal selection in the sense that it assumes the immune system to be a dynamic system by its very nature. Under the network perspective, immune cells and molecules are capable of recognizing each other and thus presenting some pattern of dynamic behavior even if not in the presence of foreign stimulation.

With this dynamical view of the immune system, it is most natural that the pioneer models of the immune network theory were based upon Ordinary Differential Equations (ODEs) as means to quantify the dynamics of individual cells and molecules in the immune system. These *continuous* models were very important for the development of the field of AIS, and also served as basis for

the proposal of *discrete* immune network algorithms. This section reviews both approaches separately. The focus will be on the specific continuous network model introduced by Farmer and collaborators (Farmer et al., 1986), and on a more generic discrete network model. The examples of application will, thus, focus on particular versions of both models.

The network approach is particularly interesting for the development of computational tools because it naturally provides an account of emergent properties such as learning, memory, self-tolerance, size and diversity of cell populations, and network interactions with the environment and the self components (de Castro and Timmis, 2002). What both models, continuous and discrete, have in common, is a general dynamics that can be summarized as follows:

$$\text{RPV} = \text{NSt} - \text{NSu} + \text{INE} - \text{DUE} \tag{6.9}$$

where RPV is the rate of population variation, NSt is the network stimulation, NSu is the network suppression, INE is the influx of new elements, and DUE is the death of unstimulated elements. Note that the first two terms on the right hand side of Equation (6.9) are related to the immune network dynamics, and the last two terms correspond to the immune metadynamics.

The network activation embodies the stimulation of a cell or molecule by another cell or molecule of the immune system or a foreign antigen, while network suppression is due to self-recognition only.

6.7.1. Continuous Immune Networks

The paper by Farmer et al. (1986) is considered the pioneer work relating the immune system with other artificial intelligence approaches (de Castro and Timmis, 2002). They described a dynamical model for the immune system based upon the immune network theory. Their model consisted of a set of coupled differential equations to quantify the dynamics of the components of the immune network.

In the artificial immune system proposed by Farmer and collaborators, the cells and molecules are represented as variable-length binary strings in a Hamming shape-space and were allowed to interact in various different alignments. Let the *epitope* (*e*), also called *idiotope* (*i*), be the portion of the antibody molecule that can be recognized by another portion of the antibody attribute string termed *paratope* (*p*). Thus, an antibody is represented by a binary string with two parts, one corresponding to the paratope and another to the epitope, as illustrated in Figure 6.21. Epitopes and paratopes are mutually exclusive, in the sense that an epitope is not a paratope, and vice-versa.

The match between two bitstrings occurs in a complementary fashion in any possible alignment, accounting for the fact that two molecules may react in more than one way. Equation (6.10) specifies the *matrix of matching specificities* $\mathbf{M} = \{m_{ij}\}$, $\forall i,j$, containing the match value of each molecule (binary attribute string) with all the other molecules in the network; the first index of m_{ij} refers to the epitope and the second to the paratope. Matrix **M** is thus a function of the affinity (Hamming distance) among the network antibodies.

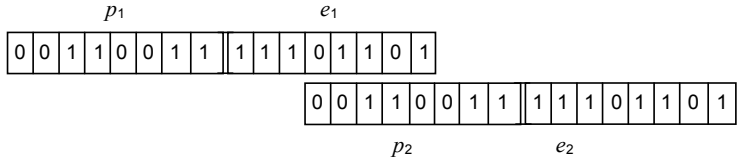


Figure 6.21: Bitstrings represent the epitope (or idiotope) and paratope of antibody molecules. (Reproduced with permission from [de Castro and Timmis, 2002], © L. N. de Castro and J. Timmis.)

$$m_{ij} = \sum_k G\left(D\left(\sum_n e_i(n+k), p_j(n) \right) - \varepsilon + 1 \right) \qquad \textbf{(6.10)}$$

where $p_j(n)$ is the n-th bit of the j-th paratope, $e_i(n)$ is the n-th bit of the i-th epitope, $D(\cdot,\cdot)$ corresponds to the Hamming distance between $e_i(\cdot)$ and $p_j(\cdot)$, given by Equation (6.3), and ε corresponds to the affinity or cross-reactivity threshold. A given alignment between a paratope and an epitope is specified by the parameter k. If a matching occurs in more than one alignment, its strength is summed to the strength of the other alignments, including the case of strings with different lengths.

Function $G(\cdot)$ measures the strength of a possible interaction between an epitope and a paratope: $G(x) = x$, if $x > 0$, and $G(x) = 0$, otherwise. For a given alignment k, G is zero if less than ε bits are complementary, while if ε or more bits are complementary, then $G = 1 + \delta$, where δ is the number of complementary bits in excess of ε.

The network dynamics takes into account *stimulation* and *suppression* events. Network stimulation is promoted by the recognition of an epitope by a paratope and results in antibody proliferation, while the antibody with the recognized paratope is suppressed and may be probabilistically eliminated from the repertoire. Assume N_1 antibody types with concentrations $\{c_1,...,c_{N_1}\}$, and N_2 antigens with concentrations $\{y_1,...,y_{N_2}\}$. The ODE that governs the dynamics of the antibody concentration, summarized in Equation (6.11), contains four parts: i) a first term corresponding to the stimulation of the paratope of an antibody type i by the epitope of an antibody type j; ii) a second term corresponding to the suppression of antibody type i when its epitope is recognized by the paratope of type j; iii) a third term that takes into account the concentration of antigens; and iv) a fourth term that models the natural death rate of antibodies.

$$\frac{dc_i}{dt} = k_1 \left[\sum_{j=1}^{N_1} m_{ji} c_i c_j - k_2 \sum_{j=1}^{N_1} m_{ij} c_i c_j + \sum_{j=1}^{N_2} m_{ji} c_i y_j \right] - k_3 c_i \qquad \textbf{(6.11)}$$

where parameter k_1 is a rate constant that depends on the number of collisions per unit time and the rate of antibody production stimulated by a collision; constant k_2 represents a possible inequality between network stimulation and suppression; and constant k_3 is the natural death rate.

Equation (6.11) describes how the concentration of antibodies varies with time. Antibodies that recognize antigens or other antibodies are stimulated and have their concentration increased, whereas antibodies that are recognized by other antibodies or do not recognize antigens are suppressed and have their concentration decreased. Very low concentration levels may lead to the death of a given cell type.

An important feature of this model is its metadynamics. That is, the repertoires of antibodies and antigens are dynamic, changing as new elements are added to or removed from the network. The continuous production of novel antibodies and their incorporation into the network provide the immune network with the ability to cope with unexpected (or unseen) antigens. To perform this, the authors used a minimum threshold on all concentrations, so that an element is eliminated when its concentration falls below the threshold. The new antibody types were generated by applying genetic operators to the bitstrings, such as crossover, inversion, and point mutations (Chapter 3).

6.7.2. Discrete Immune Networks

The main difference between the continuous immune network models and the discrete ones is that the former are based on the use of ordinary differential equations while the latter use either a difference equation or an iterative procedure to govern their dynamics. Most continuous models only account for changes in the concentration of antibodies in the network, such as the model described above. Discrete networks, however, commonly consider variations in the concentration of antibodies and in their structure (attributes) as well. This means that the elements of a discrete immune network can increase or decrease in number and can also change their attribute strings in order to improve their affinities with antigens. Finally, one last difference between them is that most continuous models were devised as means to simulate the immune network instead of as tools for problem solving. Most discrete immune networks, by contrast, are targeted at solving particular problems.

In the immune network theory introduced by Jerne (1974) the immune system is composed of a dynamic network whose behavior is perturbed by foreign antigens. Thus, there are basically two levels of interactions in the network: 1) the interaction with the foreign antigens, which correspond to the external environment; and 2) the interaction with other network elements, which represents the internal environment of the organism. As the foreign stimulation disturbs the network, it has to self-organize so as to cope with the perturbation. After the network self-organization, specific groups of cells and molecules correspond to specific antigenic groups.

Although there are several proposals of artificial immune networks in the literature, this section describes a single one. The description that follows is based on the discrete immune network proposed by de Castro and Von Zuben (2001), named aiNet (Artificial Immune Network). Other proposals that the students are strongly encouraged to look at are available in Timmis et al. (2000), Neal (2003), and in the comparative work performed by Galeano et al. (2005).

In aiNet, it is assumed a population **S** of antigens to be recognized and a randomly initialized set **P** of cells in the network. Each network element corresponds to an antibody molecule; there is no distinction between the antibody and the B-cell. A real-valued vector in an Euclidean shape-space is used to represent antibodies and antigens. Each antigenic pattern is presented to each network cell and their affinity is determined using the Euclidean distance (Equation (6.1)). A number of high affinity antibodies is selected and reproduced (*clonal expansion*) based on their affinity with the antigen: the higher the affinity, the higher the number of clones and vice-versa. The clones generated undergo somatic mutation inversely proportional to their antigenic affinity: the higher the affinity, the lower the mutation rate. A number of high affinity clones is selected to be maintained in the network, constituting what is defined as a clonal memory.

The affinity among the remaining antibodies is determined. Those antibodies whose affinity is less than a given threshold are eliminated from the network in a process known as *clonal suppression*. All antibodies whose affinity with the antigen is less than a given threshold are also eliminated from the network (*natural death rate*). Additionally, a number of new randomly generated antibodies are incorporated into the network (*metadynamics*). The remaining antibodies are incorporated into the network and their affinity with the existing antibodies is determined. All but one antibody whose affinity is less than a given threshold are eliminated.

The aiNet learning algorithm can be summarized as follows (de Castro and Timmis, 2002):

1. *Initialization*: create an initial random population **P** of network antibodies.
2. *Antigenic presentation*: for each antigenic pattern **s**, do:
   - 2.1 *Clonal selection and expansion*: for each network element, determine its affinity with the antigen presented. Select a number n_1 of high affinity elements and reproduce (clone) them proportionally to their affinity.
   - 2.2 *Affinity maturation*: mutate each clone inversely proportional to affinity. Re-select a number n_2 of highest affinity clones and place them into a clonal memory set.
   - 2.3 *Metadynamics*: eliminate all memory clones whose affinity with the antigen is less than the affinity (cross-reactivity) threshold ε.
   - 2.4 *Clonal interactions*: determine the network interactions (affinity) among all the elements of the clonal memory set.
   - 2.5 *Clonal suppression*: eliminate those memory clones whose affinity with each other is less than a prespecified suppression threshold σ_s.
   - 2.6 *Network construction*: incorporate the remaining clones of the clonal memory into the network.
3. *Network interactions*: determine the similarity between each pair of network antibodies.
4. *Network suppression*: eliminate all network antibodies whose affinity with each other is less than a prespecified suppression threshold σ_s.

5. *Diversity*: introduce a number n_3 of new randomly generated antibodies into the network.
6. *Cycle*: repeat Steps 2 to 5 until a prespecified number of iterations (`max_it`) is reached.

```
procedure [M] = aiNet(S,max_it,n1,n2,n3,σs,ε)
    initialize P
    t ← 1
    while t <= max_it do,          //for max_it iterations
       vet_permut ← randperm(N)  //permutations of N
       for j from 1 to M do,      //S contains M antigens
            //select the index i of antigen si to be presented
            i ← vet_permut(j)      //present antigens randomly
            aff ← affinity(si,P)
            P1 ← select(P,n1)      //n1 highest affinity
            C ← clone(P1,aff)      //proportional to affinity
            C1 ← mutate(C,aff)     //inversely proportional
            aff ← affinity(si,C1)//affinity of clones with s
            Ms ← select(C1,n2)     //clonal memory for s
            for each m of Ms do,
               //death of unstimulated cells (metadynamics)
               if aff(Ms) > ε then,
                    remove m from Ms
               end if
               //clonal suppression
               clonal_affm ← affinity(m,Ms)
               if clonal_affm < σs then,
                    remove m from Ms
               end if
            end for
            //network construction
            M ← insert(P,Ms)       //introduce Ms in M
       end for
       //network suppression
       for each m of M do,
            affm ← affinity(M,m)  //network interactions
            if aff(M) < σs then,   //network suppression
               remove m from M
            end if
       end for
       R ← rand(n3)                //n3 new random Abs
       P ← insert(M,R)             //diversity introduction
       t ← t + 1
    end while
end procedure
```

Algorithm 6.8: The aiNet adaptation procedure.

Let **S** be the set of M patterns (antigens) to be recognized, `max_it` the number of iterations to be performed, n_1 the number of high affinity antibodies to be selected for reproduction, n_2 the number of high affinity antibodies to become clonal memory, n_3 the number of new antibodies to be introduced into the population, σ_s the suppression threshold, and ε the cross-reactivity threshold. The suppression threshold is responsible for eliminating antibodies that are too similar to other antibodies in the network, and the cross-reactivity threshold eliminates all antibodies that do not have a minimal degree of affinity with the antigenic pattern presented. Algorithm 6.8 summarizes the adaptation procedure of the aiNet algorithm. It returns a matrix **M** of memory antibodies.

6.7.3. Selected Applications from the Literature: A Brief Description

This section illustrates the use of both types of immune networks: continuous and discrete. The potential of application of a continuous immune network model is illustrated by listing the main features of an AIS system to recommend videos, and the discrete network model, whose pseudocode was described in Algorithm 6.8, will be used to perform data compression and clustering.

A Recommender System

Based on information about the user's profile, a *recommender* or *recommendation system* is the one designed to predict items (e.g., books, movies, music, etc.) that this user may be interested in. Recommender systems based on *collaborative filtering* methods make predictions about the interests of (and recommendations to) a user by gathering information from many similar users, sometimes called neighbors.

Cayzer and Aickelin (2005) used a modified version of Farmer's continuous immune network model to the task of movie recommendation by collaborative filtering. The users were encoded as a set of 2-tuples, $user = \{\{id_1,score_1\}, \{id_2,score_2\}, \ldots, \{id_n,score_n\}\}$, where id_i is the identifier of movie i, $score_i$ is the score given by the user to movie i, and n is the number of movies for recommendation. The main aspects to be decided in their proposal were: i) how to select the neighborhood of a given user; and ii) how to predict the rating of a movie by this user.

The authors chose the Pearson correlation coefficient to compare two users a and e, which is basically the dot product between the two normalized vectors:

$$p_{ae} = \frac{\sum_{i=1}^{N}(a_i - \bar{a})(e_i - \bar{e})}{\sqrt{\sum_{i=1}^{N}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{N}(e_i - \bar{e})^2}} \tag{6.12}$$

where a_i is the vote of user a for movie i, e_i is the vote of user e to movie i, $\bar{a}$ is the average vote of user a over all movies including the overlapping votes, $\bar{e}$ is the average vote of user e over all movies including the overlapping votes, and N is the number of overlapping votes (i.e., movies for which both a and e have voted). In case there is no overlapping vote or the denominator of Equation (6.12) is zero, $p_{ae} = 0$. Their neighborhood selection scheme corresponded to

choosing the best k absolute correlation scores, where k is the neighborhood size.

For the immune network predictor, a user for whom to make a prediction was encoded as the antigen **Ag** and a network of antibodies was initialized (the other users were the antibodies). At each step (iteration) an antibody's concentration was increased in a direct proportion to its matching score to the antigen (measured via the Pearson correlation coefficient), decreased in a direct proportional to its match in relation to other antibodies in the network, and decreased due to aging. To account only for antigenic stimulation, antibody suppression and aging, the authors simplified Equation (6.11), turning it into

$$\frac{dx_i}{dt} = k_1 p_i x_i y_j - \frac{k_k}{n} \sum_{j=1}^{n} p_{ji} x_i x_j - k_3 x_i \qquad \textbf{(6.13)}$$

where k_1, k_2 and k_3 are constants that rate antigenic stimulation, network suppression and natural death rate, respectively. Parameter p_i is the correlation between antibody i and the single antigen (user for whom to make the prediction), x_i is the concentration of antibody i, y is the concentration of the antigen, p_{ij} is the correlation between antibodies i and j, and n is the number of antibodies in the network. Note that the authors only use network suppression; no network stimulation is accounted for.

The rating r_i of a given movie i by the user a is determined by employing a weighted average of the neighborhood $k = N$ of the user, taken to be the whole network of antibodies. Let w_{ae} be the weight between users a and e, p_{ae} be the correlation score between a and e, and x_e be the concentration of the antibody corresponding to the user e, then

$$w_{ae} = p_{ae} x_e$$

$$r_i = \bar{a} + \frac{\sum_{e \in N} w_{ae} (e_i - \bar{e})}{\sum_{e \in N} w_{ae}}$$

The authors performed a number of experiments on a public dataset with their AIS, taking and not taking into account the immune network effects. As a comparative approach they used the simple Pearson (SP) predictor (Herlocker et al., 1999). The results showed that the AIS without network interactions performed similarly to the SP method in terms of prediction, but presented superior recommendation accuracy when network interactions were added.

Data Compression and Clustering

To illustrate the performance of the aiNet when applied to data compression and clustering, consider the two-spirals (SPIR) benchmark problem illustrated in Figure 6.22(a). This data set is composed of 190 samples in $\Re^2$ and aims at testing the aiNet capability to detect non-linearly separable clusters. Note that though the samples are labeled in the picture they are unlabeled for the aiNet. The following parameters were adopted in this simulation: $g = 40$, $n_1 = 4$, $n_2 = n_3 = 10$, $\alpha = 0.07$, and $\beta = 1.0$.

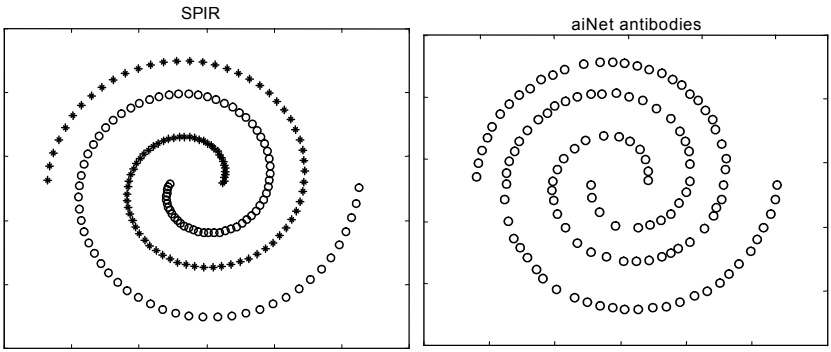


Figure 6.22: aiNet applied to the two spirals (SPIR) problem. (a) Input patterns. (b) Antibodies generated by the aiNet adaptation procedure presented in Algorithm 6.8.

Figure 6.22(b) depicts the final set of antibodies generated by the aiNet. In this case, the resultant memory matrix was composed of $m = 121$ antibodies, corresponding to a $CR = 36.32\%$ reduction in the size of the sample set. Note that the compression was superior in regions where the number of data (amount of redundancy) is larger, i.e., the centers of the spirals.

The next important question that remains is related to how to define the clusters of this network. The simplest answer to this question is to connect all cells in the network and remove those connections greater than a given threshold. Several authors still use this idea. Another simple, but more clever form of separating clusters of a trained aiNet, is to use the concept of the *minimal spanning tree* (MST). The MST of the resultant network antibodies is built and a criterion to eliminate some edges of the MST is used in order to define some clusters in the network (Appendix B.4.4).

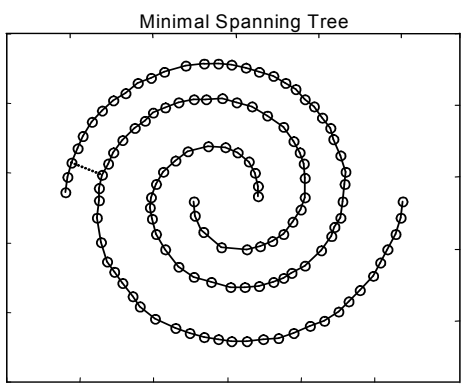


Figure 6.23: Minimal spanning tree for the antibodies produced by the aiNet adaptation procedure.

Building the MST and Identifying Clusters

For the two spirals problem under study, the resultant MST is illustrated in Figure 6.23. Assuming a factor $r = 2$, the MST is pruned in the dashed edge and the two clusters corresponding to the clusters observed in the original data set are identified.

6.8 FROM NATURAL TO ARTIFICIAL IMMUNE SYSTEMS

Table 6.1 summarizes a possible interpretation of (theoretical) immunology into the domain of artificial immune systems. For instance, cells and molecules are usually represented by attribute strings in a specific shape-space, affinity corresponds to the degree of match between strings or recognition between molecules, and so on, as summarized in Table 6.1.

Artificial immune systems borrow several ideas from the biological immune system. Most algorithms to date are inspired by some sort of immune principle, process or theoretical model. In some cases, such as with clonal selection algorithms, it is evident that the immune algorithms are either hybrids with evolutionary algorithms or can be viewed as such. However, the immune algorithms still have some particular features, such as distributivity, double plasticity, anomaly detection and diversity, that give them their identity and, more importantly, that perhaps would never had been introduced without insights from immunology.

Table 6.1: Interpretation from the immunological terminology into the computational domain of AIS.

Biology (Immunology)	**Artificial Immune Systems**
Cells and molecules	Data structure (attribute strings)
Affinity	Degree of match (interaction) between data structures
Fitness	Quality of a data structure for tasks that do not involve quantifying the degree of interaction between data structures
Bone-marrow models	Used to generate data structures
Affinity function	Quantify affinities
Somatic hypermutation	Used to introduce or maintain population diversity or genetic variation
Affinity maturation	Promotes learning (adaptation) through somatic hypermutation and selection
Clonal selection	Describes how the immune cells and molecules interact with antigens
Negative selection	Generates set(s) of nonself detectors for anomaly detection
Immune network	Performs the dynamics and metadynamics of the system structured in a network-like manner

6.9 SCOPE OF ARTIFICIAL IMMUNE SYSTEMS

One of the most remarkable features of the field of artificial immune systems is its broad applicability. There are AIS applied to domains such as:

- Machine-learning and pattern recognition.
- Anomaly detection and the security of information systems.
- Data analysis (knowledge discovery in databases, clustering, etc.).
- Agent-based systems.
- Scheduling.
- Autonomous navigation and control.
- Search and optimization.
- Artificial life.

There are a few survey papers, reports, and book chapters that can be used by the reader to find information on where and when AIS have been developed and applied. Chapter 4 and Chapter 6 of the book by L. N. de Castro and J. Timmis (2002) survey a vast amount of works on AIS. Y. Ishida (1996), D. Dasgupta and Attoh-Okine (1997, 1999), L. N. de Castro and F. Von Zuben (2000), Dasgupta et al. (2001), L. N. de Castro (2004), and U. Aickelin et al. (2004) all have published survey papers, reports or bibliographies of artificial immune systems and their applications.

6.10 SUMMARY

In the previous chapter we have seen that the cells (neurons) composing the nervous system are connected with one another via small junctions called synapses. The modulation of these synapses is believed to be the basic means by which an ensemble of nerve cells, called neural network, adapts and learns. The nervous system is known to be structured in a network-like manner. In the case of the immune system, however, three main perspectives can be found in the literature (de Castro, 2003). The first one assumes an immune system composed of discrete sets of cells and molecules targeted at performing self/nonself discrimination. The second one suggests an immune system composed of interconnected sets of cells and molecules, resulting in a network of affinities within the immune system. The last perspective is that the immune system works in concert with many other bodily systems, mainly the nervous and endocrine systems, in order to main *homeostasis* (i.e., the body's internal equilibrium state).

One important feature of a great number of artificial immune systems is the presence of adaptability in two levels: structural adaptability and parametric adaptability. In the first level, the architecture of the system is allowed to adapt to the environment, and in the second level, the parameters or attributes are also varied with time. This double-plasticity leads to high degrees of flexibility and robustness of AIS.

This chapter reviewed a framework to design artificial immune systems, including how to create abstract models of immune cells and molecules, quantifying their interactions, and using adaptation procedures. It became clear that there

are much more to artificial immune systems than this single chapter could have reviewed. Other algorithms are available and some of these have been discussed in the literature cited throughout the chapter.

6.11 EXERCISES

6.11.1. Questions

1. What is the difference between a vaccine and a serum?
2. When the immune system mistakenly recognizes self as being nonself, it mounts an immune response against self, thus causing the so-called auto-immune diseases. Name three examples of auto-immune diseases and explain their main causes.
3. If the immune system is capable of differentiating between self and nonself, why doesn't it reject the fetuses?
4. It is known that there must be some compatibility between the mother's and the father's blood type. What is the relationship between blood type compatibility in parents and the immune system?
5. The self/nonself discrimination issue occurs differently for B-cells and T-cells. Explain the positive selection of B-cells and T-cells, and the negative selection of B-cells and T-cells. How are these concepts related with the immune tolerance?
6. Allergies are known to be related to a mal-functioning of the immune system. Explain what allergies are and why they are caused.

6.11.2. Computational Exercises

1. Use a bone marrow algorithm to define genes for gene libraries to be used to generate the initial population of a genetic algorithm to solve the TSP problem presented in Chapter 3 and Chapter 5 (Figure 6.24). Assume the following structure for the gene libraries:

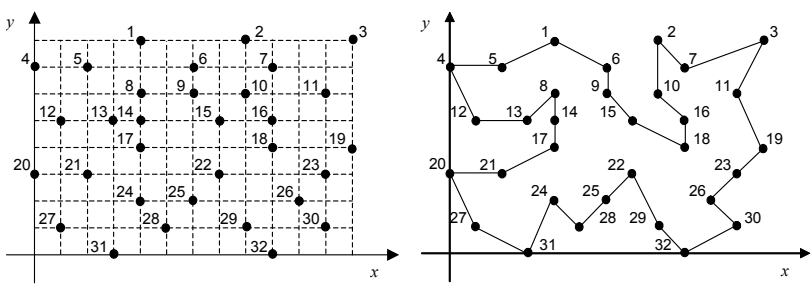


Figure 6.24: Simple TSP with 32 cities. The cities are placed on a regular grid on the *x*-*y* plane, in which each point (•) represents a city, the number next to each city is its index, and each square on the grid corresponds to one unit of distance (uod - e.g., Km).

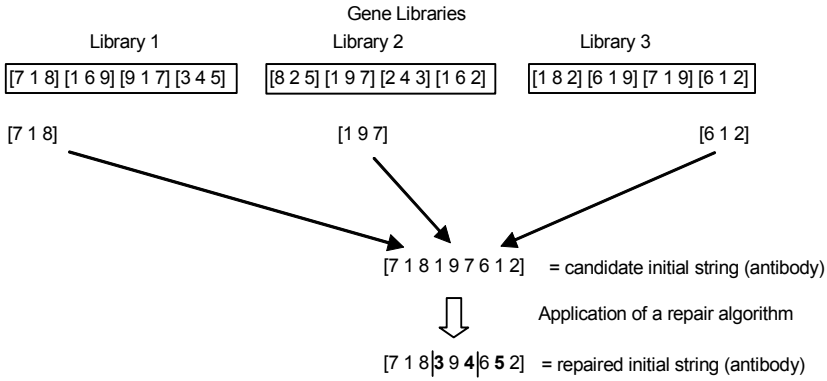


Figure 6.25: Generation of initial antibodies (chromosomes) and application of a repair algorithm.

Gene length $L_g = 4$, number of libraries $n = 8$, and library length (number of genes in each library) $L_l = 4$. As one gene from each library will be selected, the total chromosome length is $L = L_g \times n = 4 \times 8 = 32$, that corresponds to the number of cities in a tour.

Each gene will be defined as a sequence of four cities known to be part of an optimal route. For example, gene one from library A, g_{1A}, can be defined as $g_{1A} = [19\ 23\ 26\ 30]$, gene two from library A, g_{2A}, can be defined as $g_{2A} = [32\ 29\ 22\ 25]$, and so on.

It is important to note that in some (maybe most) cases, one or more elements of a gene will be repeated and the index of some cities will not appear in the final chromosome generated. Thus, a repair algorithm must be used in order to generate permutations of L integers while maintaining most of the genes intact. This is because the TSP problem has the constraint that no city can be visited more than once. To illustrate this problem and one possible way of solving it, consider the example presented in Figure 6.25.

In the example of Figure 6.25, the concatenation of genes from the libraries generated a chromosome (route) with some cities repeated. To fix this candidate string, a repair algorithm that seeks for repeating and missing numbers is used, and when a repetition occurs, this repeated number is exchanged by one number from the sequence $[1..L]$ that has not appeared yet.

Implement this bone marrow model to define an initial population of chromosomes to be used in an evolutionary algorithm to solve the TSP problem illustrated. Compare the performance of the algorithm with this type of initialization procedure and with the random initialization used in Project 1, Section 3.10.4.

2. The bone marrow model using gene libraries, described in Section 6.4, can also be used as a recombination scheme in evolutionary algorithms in place of the standard crossover operation, as follows.

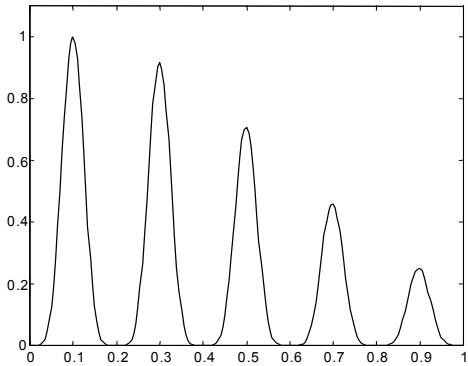


Figure 6.26: Graph of the function $g(x) = 2^{-2((x-0.1)/0.9)^2}\left(\sin(5\pi x)\right)^6$ to be maximized.

At each generation, create libraries by taking sequences (attribute strings) of each individual chromosome from the population as genes to compose the libraries. The chromosomes from which the sequences of bits are going to be taken can be chosen using heuristics, such as their fitness value. If an individual from the GA is selected for recombination with a given probability *pc*, then this individual (or part of it) can be replaced by another one generated using the gene libraries.

Modify the standard genetic algorithm with binary representation to support the recombination scheme based on gene libraries, apply it to the function optimization problem described in Section 3.3.3 and discuss the results obtained.

Function to be optimized: $g(x) = 2^{-2((x-0.1)/0.9)^2}\left(\sin(5\pi x)\right)^6$ (see Figure 6.26)

The problem corresponds to finding **x** from the range [0..1] which maximizes $g(\mathbf{x})$, i.e., to find **x*** such that $g(\mathbf{x}^*) \geq g(\mathbf{x})$, $\forall\ \mathbf{x} \in [0..1]$.

3. Given the data set illustrated in Figure 6.27, use the negative selection algorithm to generate a set of $N = 1{,}000$ detectors that recognize everything but the self patterns. Assume a cross-reactivity threshold $\varepsilon = 72$, corresponding to 60% of the length of each pattern. The affinity measure is given by *Affinity* $= L - D$, where L is the length of the strings ($L = 120$), and D is the Hamming distance between two strings (Equation (6.3)).

Figure 6.27: Set of input patterns (self set **S**).

Evaluate the set of detectors generated by randomly introducing noise in the samples, with various noise levels from 5% to 50%, and monitoring for the presence of unknown patterns. (Patterns with more than 40% of noise should be detected as nonself.)

4. Use the Animals data set, depicted in Table 6.2, as a self set to generate 1000 detectors using the binary negative selection algorithm.

   Use the nonself set depicted in Table 6.3 to calculate the detection rate (*DR*) and the false alarm rate (*FAR*), described in Equation (6.6) and Equation (6.7), respectively, of the 1000 detectors generated.

   Investigate the influence of different affinity measures (Hamming, *r*-contiguous and *r*-chunks) and affinity thresholds ε for the binary negative selection algorithm.

5. Apply the optimization version of CLONALG to the function optimization problem presented in Exercise 2. Assume both cases: a binary Hamming shape-space and a real-valued shape-space with Gaussian mutation (Section 3.3.3).

6. Implement the clonal selection algorithm CLONALG and apply it to the pattern recognition task of Section 6.6.2.

7. Apply Forrest's clonal selection algorithm (without crossover) to the problem above. Compare the performance of both CLONALG and Forrest's algorithm when applied to pattern recognition.

Table 6.2: Set of animals (names and their attributes) to be used to generate the detectors.

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
Is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has	Two legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	Four legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 6.3: Set of nonself animals to evaluate detectors.

		Swan	Penguin	Pterodactyls	Bear	Sheep	Pig	Tyrannosaurus	Snail	Snake	Shark	Dolphin	Eagle	Rooster
Is	Small	0	0	0	0	0	0	0	1	1	0	0	0	1
	Medium	0	1	0	0	1	1	0	0	0	0	0	1	0
	Big	1	0	1	1	0	0	1	0	0	1	1	0	0
Has	Two legs	1	1	1	0	0	0	0	0	0	0	0	1	1
	Four legs	0	0	0	1	1	1	1	0	0	0	0	0	0
	Hair	0	0	1	1	0	1	1	0	0	0	0	0	0
	Hooves	0	0	0	0	1	1	0	0	0	0	0	0	0
	Mane	0	0	0	0	0	0	0	0	0	0	0	0	0
	Feathers	1	1	0	0	0	0	0	0	0	0	0	1	1
Likes to	Hunt	0	1	1	1	0	0	1	0	1	1	0	1	0
	Run	1	0	1	1	1	1	1	0	0	0	0	0	0
	Fly	0	0	1	0	0	0	0	0	0	0	0	1	0
	Swim	1	1	0	1	0	0	0	1	1	1	1	0	0

8. Apply the aiNet algorithm to the data set presented in Section 5.6.2 (Exercise 3). Make use of the MST edge inconsistency criterion in order to separate the network clusters and thus identify the clusters of the original data set. Use the same parameters as those used to solve the SPIR problem, including the factor $r = 2$. Compare the performance of both algorithms: ACA and aiNet.
9. Apply the aiNet to solve the chainlink problem illustrated in Figure 6.28. This data set is composed of 1000 data points in the $\Re^3$-space, arranged such that they form the shape of two intertwined 3-D rings, of whom one is extended along the *x*-*y* direction and the other one along the *x*-*z* direction. The two rings can be thought of as two links of a chain with each one consisting of 500 data points. The data is provided by a random number generator whose values are inside two toroids with radius R = 1.0 and r = 0.1.

   The following parameters can be used to solve this problem with aiNet: $g = 40$, $n_1 = 4$, $n_2 = n_3 = 10$, $\alpha = 0.15$, and $\beta = 1.0$. Note that α is the only parameter different from those used in the SPIR problem.

6.11.3. Thought Exercises

1. How would you compare philosophically the clonal selection algorithms with the evolutionary algorithms? Hint: look at the basic steps and procedures of both theories.

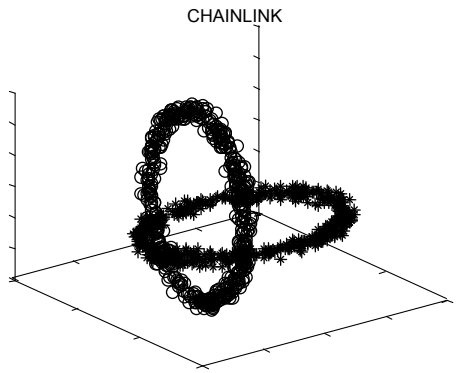


Figure 6.28: CHAINLINK problem with 1000 patterns.

2. How would you compare the generation of antibodies from gene libraries with a crossover procedure discussed in evolutionary algorithms?
3. Can you use a bone marrow model based upon gene libraries to initialize weight vectors for neural networks? If yes, how and what would be the benefits? If not, justify your answer.
4. What are the main differences between immune and neural networks? Tip: look at their design principles.

6.11.4. Projects and Challenges

1. Extend the clonal selection algorithm presented in Algorithm 6.6 to incorporate an affinity threshold ε. Write a pseudocode for the modified algorithm and test it with the data set illustrated in Figure 6.29. This data set is composed of 10 characters with resolution 20×20.

   The incorporation of a cross-reactivity threshold in the affinity measure of CLONALG leads to the possibility of having one single memory cell to represent groups of similar antigen patterns, and thus incorporates explicitly the notion of generalization in the AIS.
2. Implement the real-valued negative selection algorithm together with a multi-layer perceptron network to solve the breast cancer diagnosis experiment discussed in Section 6.5.3.

Figure 6.29: Data set to be used to test the generalization capability (cross-reactivity) of the extended CLONALG algorithm.