A Fractal Immune Network

Peter J. Bentley[1] and Jon Timmis[2]

[1] Department of Computer Science, University College London. UK
p.bentley@cs.ucl.ac.uk
http://www.cs.ucl.ac.uk/staff/p.bentley/
[2] Computing Laboratory, University of Kent, Canterbury. UK.
j.timmis@kent.ac.uk

Abstract. Proteins are the driving force in development (embryogenesis) and the immune system. Here we describe how a model of proteins designed for evolutionary development in computers can be combined with a model of immune systems. Full details of a prototype system are provided, and preliminary experiments presented. Results show that evolution is able to adjust the mapping between input data and antigens and cause useful changes to the subnetworks formed by the immune algorithm.

1 Introduction

Human development (embryogenesis) is a highly evolved network of cellular and chemical interactions, which somehow manages to build and maintain our bodies throughout our lives (Wolpert et al, 2001). One of the processes built and maintained by development is our immune system. The human immune system is also a highly evolved network of cellular and chemical interactions, which somehow learns, predicts and correctly protects our bodies from invading pathogens and internal malfunctions.

Clearly, the features of immune systems are highly related to development. Indeed, immune systems can be considered to be a specialized form of development, which focuses on the removal of unwanted elements from us. Both work in much the same ways: genes produce proteins which control other genes, and determine the function of cells. In development, our genetic program causes cells to divide, move, differentiate, extrude substances, and signal other cells using special proteins. In immune systems, our genetic program causes immune cells to be created, to respond to signals from other damaged or infected cells, and to send and receive signals from each other.

Throughout, proteins are the driving force. Produced by genes, they activate or suppress other genes, they determine the fate of cells, and they act as signals, enabling cells to communicate with each other. It is the shape of these proteins that determines their function. Through a complex bio-chemical process of protein folding, every protein has a unique morphology which enables it to diffuse between cells, interact with other proteins, attach to receptors on cell walls, interact with genes, and perform thousands of other actions.

In an attempt to harness more of the capabilities of proteins, in this work we describe how a model of proteins based on fractals, designed for evolutionary development in computers, can be combined with a model of immune systems. The *fractal immune network* maps data items to *fractal antigens*, creates *fractal recognition spaces* (similar to Artificial Recognition Balls) in dynamic networks, and forms all network links by emission and reception of *fractal cytokines*. The system is essentially a reconfigurable clusterer – the networks of fractal recognition spaces can be radically changed by changing the mapping from data to fractal antigens. The system evolves this mapping according to a fitness function, thus automatically providing desirable clusters and data classification, regardless of the data.

2 Fractal Proteins

Other work by the first author (Bentley, 2004; Kumar and Bentley, 2003) has focused on biologically plausible models of gene regulatory networks (GRNs) in the context of development (embryogenesis). In such models, genes define proteins, which trigger or suppress (i.e., *regulate*) the activation of the genes, causing dynamic, non-linear regulatory networks to form. By evolving the genes using a genetic algorithm, the resulting networks can be linked to sensors and functions and can be used for tasks such as function regression (Bentley 2004) or robot control (Bentley 2003a). In the context of a full developmental model, GRNs specify how cells divide, grow, differentiate and die, in order to produce a larger, more complex multicellular solution (Kumar and Bentley, 2003).

The recent work of (Bentley, 2004, 2003a,b) developed a new model of proteins to overcome various difficulties with previous models. Here, genes are expressed into *fractal proteins* – subsets of the Mandelbrot set that can interact and react according to their own fractal chemistry. The motivations behind this work are extensive and can briefly be listed as follows: (Further motivations and discussions on fractal proteins are provided in (Bentley, 2004, 2003b).)

1. Natural evolution extensively exploits the complexity, redundancy and richness of chemical systems in the design of DNA and the resulting developmental systems in organisms. Providing a computer system with genes that define fractal proteins gives the system complexity, redundancy and richness to exploit.
2. It is extremely difficult and computationally intensive to model natural chemical systems accurately in an artificial chemistry. Fractal proteins have many of the same properties as natural proteins, without any modelling overheads.
3. A fractal protein (with the infinite complexity of the Mandelbrot set) can be defined by just three genes.
4. The "fractal genetic space" is highly evolvable – a small change to a gene produces a small change to the fractal protein, while the self-similarity of fractals ensures that any specific shape can be found in an infinite number of places.
5. When fractal proteins are permitted to interact according to their morphologies, a hugely complex (and eminently exploitable) fractal chemistry emerges naturally.
6. Calculating subsets of Mandelbrot sets is *fast* so there is little overhead.

2.1 Mandelbrot Set

Given the equation $x_{t+1} = x_t^2 + c$ where x_t and c are imaginary numbers, Benoit Mandelbrot wanted to know which values of c would make the length of the imaginary number stored in x_t stop growing when the equation was applied for an infinite number of times. He discovered that if the length ever went above 2, then it was unbounded – it would grow forever. But for the right imaginary values of c, sometimes the result would simply oscillate between different lengths less than 2.

Mandelbrot used his computer to apply the equation many times for different values of c. For each value of c, the computer would stop early if the length of the imaginary number in x_t was 2 or more. If the computer hadn't stopped early for that value of c, a black dot was drawn. The dot was placed at coordinate (m, n) using the numbers from the value of c: ($m + ni$) where m was varied from –2.4 to 1.34 and n was varied from 1.4 to -1.4, to fill the computer screen. The result was the infinite complexity of the "squashed bug" shape we know so well today. (Mandelbrot, 1982)

2.2 Defining a Fractal Protein

A fractal protein is a finite square subset of the Mandelbrot set, defined by three codons (x,y,z) that form the coding region of a gene in the genome of a cell. Each (x, y, z) triplet is expressed as a protein by calculating the square fractal subset with centre coordinates (x,y) and sides of length z, see fig. 1 for an example. In this way, it is possible to achieve as much complexity (or more) compared to natural protein folding in nature.

In addition to shape, each fractal protein represents a certain *concentration* of protein (from 0 meaning "does not exist" to 200 meaning "saturated"), determined by protein production and diffusion rates.

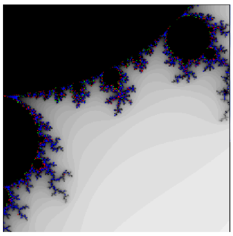

Fig. 1. Example of a fractal protein defined by (x = 0.132541887, y = 0.698126164, z = 0.468306528)

2.3 Fractal Chemistry

Cell cytoplasms and cell environments usually contain more than one fractal protein. In an attempt to harness the complexity available from these fractals, multiple proteins are merged. The result is a product of their own "fractal chemistry" which naturally emerges through the fractal interactions.

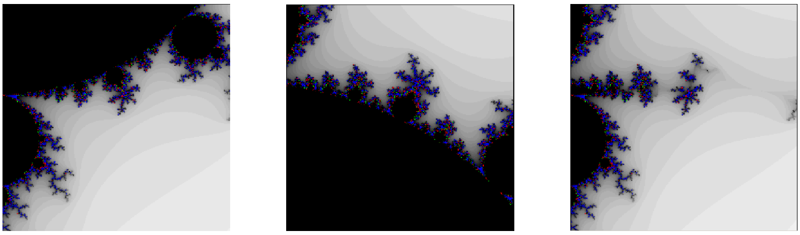

Fig. 2. Two fractal proteins (left and middle) and the resulting merged fractal protein combination (right).

Fractal proteins are merged (for each point sampled) by iterating through the fractal equation of all proteins in "parallel", and stopping as soon as the length of any is unbounded (i.e. greater than 2). Intuitively, this results in black regions being treated as though they are transparent, and paler regions "winning" over darker regions. See fig 2 for an example.

2.4 Fractal Development

Figure 3 illustrates the representation. Although not used here, fig. 4 provides an overview of the algorithm used in (Bentley, 2004, 2003a,b) to develop a phenotype from a genotype. Note how most of the dynamics rely on the interaction of fractal proteins. Evolution is used to design genes that are expressed into fractal proteins with specific shapes, which result in developmental processes with specific dynamics.

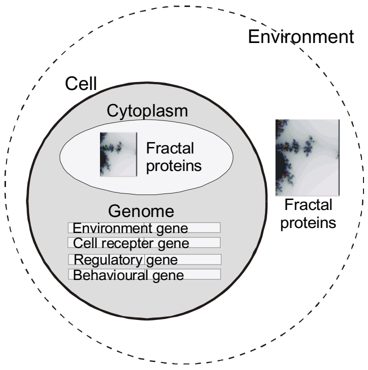

Fig. 3. Representation using fractal proteins.

```
FRACTAL DEVELOPMENT

For every developmental time step:
    For every cell in the embryo:
        Express all environment genes and
        calculate shape of merged environment fractal proteins

        Express cell receptor genes as receptor fractal proteins
        and use each one to mask the merged environment proteins
        into the cell cytoplasm.

        If the merged contents of the cytoplasm match a promoter
        of a regulatory gene, express the coding region of the gene,
        adding the resultant fractal protein to the cytoplasm.

        If the merged contents of the cytoplasm match a promoter of a
        behavioural gene, use coding region of the gene to specify a
        cellular function.

        Update the concentration levels of all proteins in the cytoplasm.
        If the concentration level of a protein falls to zero, that protein
        does not exist.
```

Fig. 4. The fractal development algorithm

3 Immune Networks

How the immune system remembers encounters with antigenic material has been a question immunologists have been asking for many years. A number of theories abound, from antigenic retention, to immune networks. A theory first proposed by Jerne (1974) suggested that B-cell memory was maintained via an Idiotypic network. This theory attempts to explain how B-cells survive even in the absence of antigenic stimulus. It is proposed that this is achieved by stimulation and suppression between B-cells via a network communicating via idiotypes on regions called paratopes; these are located on B-cell receptors. An idiotope is made up of amino acids within the variable region of an antibody or T-cell. The network acts as a self-organising and self-regulatory mechanism that captures antigenic information. Notable work in (Farmer et al. 1986) further explored the immune network theory and created a simple model of the Idiotypic network. This theory was further extended by (Perelson 1989). A network of B cells is thus formed and highly stimulated B cells survive and less stimulated B cells are removed from the system, as result is a meta-stable memory structure that exhibits interesting dynamics and properties.

Although it is now known that networks in the immune system are rather more complex than described in the immune network theory as described above, the ideas have led to a wide variety of immune-inspired algorithms to be created over the past few years (de Castro and Von Zuben, 2001), (Timmis and Neal, 2001), (de Castro and Timmis, 2002a) and Neal, 2003, to name a few.

3.1 A Meta-stable Artificial Immune Network

Work in Neal (2003) was based on an early artificial immune network model (AINE) devised for data clustering (Timmis and Neal 2001). The algorithm proposed in Neal (2003) allows for the creation of a network structure that captures the patterns (or clusters) contained within a constant input data stream. This algorithm has a number of attractive properties such as being able to identify new clusters that appear in the data stream, allows patterns to be remembered for long period of time without the need for re-enforcement from the input pattern and operates an adaptive memory mechanism that allows the network to be dynamic. It is this dynamic property of the network, that we wish to capture and augment with the use of fractal proteins. For the purposes of this paper, we will describe the system in terms of the representation or shape space of the system (coupled with an affinity function) and the algorithm used to control the population.

Representation and Shape Space

The immune system is made up of large number of B-cells and T-cells. These cells contain surface receptor molecules whose shapes are complementary to the shapes of antigens (invading material), allowing the cells to recognise the antigen and then elicit some form of immune response. Perelson and Oster (1979) first proposed the concept of *shape-space (S)*. Given that the recognition of antigens is performed by the cell receptors, shape-spaces therefore allow a quantitative description of the interactions of receptor molecules and antigens. As in the biological immune system, in a shape-space S, the degree of binding (degree of match or affinity) between an antigenic receptor or antibody **(Ab)** and an antigen **(Ag)**, is measured via regions of complementarity. Within an AIS the generalised shape of a molecule (m) of either an antibody **(Ab)** or an antigen **(Ag)**, can be represented as an attribute string (set of coordinates) $\mathbf{m = <m1, m2, ..., m_l>}$ or many other forms of attributes, ranging from integers, binary values, or other more complex structures. Within the work of Neal (2003), a simple real valued shape space was employed.

The work of (de Castro and Timmis, 2002b) proposed that this notion of shape space could be used to model an antibody and an antigen. They also stated that what ever shape is chosen, will of course affect the way in which interactions are calculated between them, i.e. how their affinity will be calculated. Given that the Ag-Ab affinity is related to distance, they stated that this could be estimated via any distance measure between two strings or vectors. In the case of Neal (2003) the Euclidean distance measure was employed.

The Algorithm

Input data is presented continuously to the network, with each data item being presented to each Artificial Recognition Balls (ARB) in the network. Experiments demonstrated that once the network had captured the patterns in the data stream, then the data stream could be removed and the patterns would remain for a period of time, due to the network interactions of stimulation and resource allocation employed in the network (Neal, 2003).

Network Initialisation. The algorithm is initialised as a network of ARB objects. Each ARB represents a data vector, a stimulation level and a number of resources (used to control lifespan of the object). Links between ARBs are created if they are below the Network Affinity Threshold (NAT), which is the average Euclidean distance between each item in the data set. The initial network is a random selection of the data set to be learnt (or a set of randomly initialised vectors), the remainder makes up the antigen training set.

Stimulation. ARBs maintain a record of stimulation. In effect, this records how well the ARB matches a certain training data item and how well it matches neighbors within the network structure. The idea being that similarly occurring patterns will reinforce each other. It is worthy of note, that no suppression element is used in the equation, unlike the work of (Timmis and Neal, 2001). This is due to the fact, that with the resource allocation mechanism employed (see below) by the author, the suppression is no longer required.

Expansion. The system does not perform cloning or mutation in the same sense as many AIS algorithms (or indeed the natural immune system). In order to allow the network to grow, the affinity of an ARB to a training data item is required to be over a certain threshold, the *affinity threshold*. If this is the case, then a new ARB is created in the location of the antigen. This is effectively adding the new cell to the network representation of self.

Resource Allocation. Each ARB is assigned an initial number of resources. The resource level is used to indicate when the death of an ARB should occur, as if the resource level falls below a certain defined threshold, then the ARB is removed from the system: in effect, this is the algorithms population control mechanism. Simply put, each ARB is assigned a local number of resources, which is proportional to the stimulation of the ARB, minus a certain geometric death rate. This enables well-stimulated ARBs to survive in the network, and poorly stimulated ARBs to be removed.

4 A Fractal Immune Network

The concept proposed and partially investigated in this paper is to combine the ideas of fractal proteins with immune networks. Since fractal proteins have been shown to enable considerable benefits to developmental models, it is suggested here that similar benefits may be gained from their use in an immune network algorithm.

The combination of these ideas is relatively straightforward. Antigens, antibodies, and cytokines are all types of protein. The immune network relies on interactions between proteins. A Fractal Immune Network, therefore uses fractals to represent these proteins, and thus the network dynamics are created by interactions between different fractal shapes. (The immune network algorithm is based on (Neal 2003).)

4.1 Data to Fractal Antigen Mapping

To enable all interactions to take place in a fractal shape space, each data item of the incoming data stream is mapped to a fractal protein, which we term a *fractal antigen* $\mathbf{Ag}^f$. Each data item (comprising 4 values) is mapped to the (x, y, z) triplet (described in section 2) using equation [1]. Once mapped, the fractal antigen becomes a finite subset of the Mandelbrot set, and is stored as a bitmap. All operations in the network then take place using bitmap processing.

Mapping data to fractal protein (x, y, z)

$$x = A_0 * \text{datum}_0 / \text{scale}_0 - B_0$$
$$y = A_1 * \text{datum}_1 / \text{scale}_1 - B_1$$
$$z = A_2 * \text{datum}_2 / \text{scale}_2 - B_2 + A_3 * \text{datum}_3 / \text{scale}_3 - B_3 \quad (1)$$

where:

datum$_i$ is the ith value of current data item
A_i, B_i are mapping coefficients, evolved by the system
(initial values for all $A_i = 1.0$, $B_i = 0.5$).
scale$_i$ is data range of values in ith column of data (assuming only positive values)

Clearly the mapping from data item to $\mathbf{Ag}^f$ is critical, so the algorithm incorporates a simple (1+1) Evolution Strategy (Bäck, 1996), which mutates the mapping coefficients every EVOLVEFREQ iterations (where EVOLVEFREQ is set to 1500 in the experiments to enable the entire data set to be presented 10 times). After mutating, the network is restarted. If the subsequent network scores a lower fitness, the coefficients are restored to their previous values before mutating again. Fitness of the network is calculated by presenting the entire dataset to the current network and using the fitness calculation shown in equation [2].

Fitness calculation for Evolution Strategy when evolving mapping coefficients:

$$F = | n_1 - class_1 | + | n_2 - class_2 | + n_3 + n_4 + nodes \qquad (2)$$

where:

n_1 is the highest number of data items from class 1 in a subnetwork p
n_2 is the highest number of data items from class 2 in a subnetwork q
(where p is not the same network as q).
n_3 is the number of data items from class 1 misclassified in subnetwork q
n_4 is the number of data items from class 2 misclassified in subnetwork p
$class_i$ is the ith class of data, see section 5.1 for an example.
nodes is the number of **FRS**s in the entire network minus five if that number is above five, zero otherwise.

4.2 Initialisation and Expansion

Initially, the network is presented with a small subset of the data (for the IRIS data used in the experiments, 10 out of 150 items are randomly picked). The data item is mapped to a fractal antigen as described above. Upon first encountering an $\mathbf{Ag}^f$, the fractal immune network algorithm creates its version of an ARB at that point, if one sufficiently similar does not already exist: this represents the initial population of the network. For this system we refer to ARBs as "Fractal Recognition Spaces" or **FRS**s. Thus, each **FRS** is defined by a core $\mathbf{Ag}^f$ (which may subsequently match one or more fractal antigens). All fractal shapes are stored as bitmaps.

After the initialization, the network runs continuously, being presented with randomly chosen data items from the entire data set, and creating more **FRS**s if needed.

4.3 Stimulation

Should any $\mathbf{Ag}^f$ match an existing **FRS** closely enough (determined by measuring whether the difference between the bitmaps is below ANTIGENMATCHINGTHRESHOLD), the **FRS** is *stimulated* according to equation [3].

Antigen stimulation S_1 calculation:

$$S_1 = 10 * (1 - d_{FRSAg} / \text{ANTIGENMATCHINGTHRESHOLD}) \qquad (3)$$

where:

S_1 is stimulation increase for **FRS** when $\mathbf{Ag}^f$ matched
d_{FRSAg} is matching distance between **FRS** and $\mathbf{Ag}^f$
ANTIGENMATCHINGTHRESHOLD is the **FRS** matching threshold (set to 500)

If required, an optional feature of this algorithm in this situation creates a new **FRS** with a probability of 0.01 in addition to stimulating the existing one. The new **FRS** is a mutation (e.g. as caused by hypermutation in immune systems), created by merging the existing **FRS** with the current $\mathbf{Ag}^f$. In this way, the core fractal protein becomes a "general" fractal antibody that will match more fractal antigens - the space of the **FRS** is widened or adjusted.

If the current $\mathbf{Ag}^f$ matches no exising **FRS**s closely enough, a new **FRS** is created as described in section 4.2.

In addition, **FRS**s in the network stimulate each other. Instead of explicitly maintaining a list of links between **FRS**s, this system makes dynamic network connections between **FRS**s, formed by the emission and reception of *fractal cytokines* $\mathbf{Ck}^f$s. To calculate the network stimulations, each **FRS** emits a $\mathbf{Ck}^f$s (simply a clone of the transmitting **FRS** bitmap) to every other **FRS**. This is bitwise masked by the receptor $\mathbf{Rc}^f$ of the receiving **FRS** (simply a clone of the receiving **FRS** bitmap) thus ensuring that two similar **FRS**s will be able to "communicate" but dissimilar **FRS**s will never receive each others' signals. If the receiving **FRS** is mature, it compares the masked $\mathbf{Ck}^f$ with itself – if the difference between the bitmaps is les than CYTOKINEMATCHINGTHRESHOLD, then the stimulation of that **FRS** is increased, as defined in equation [4].

Network stimulation S_2 calculation:

$$S_2 = d_{FRSCk} / \text{CYTOKINEMATCHINGTHRESHOLD} \quad (4)$$

where:

S_2 is stimulation increase for **FRS** when masked $\mathbf{Ck}^f$ matched
d_{FRSCk} is matching distance between **FRS** and masked $\mathbf{Ck}^f$
CYTOKINEMATCHINGTHRESHOLD is network matching threshold (set to 550)

An **FRS** is mature if its age is above MATUREAGE (set to the number of data items in the dataset) – needed when the "mutate FRS" option is activated to prevent excessive numbers of **FRS**s that match few or no $\mathbf{Ag}^f$s from overrunning the network.

Each iteration, the stimulation for each **FRS** is calculated, summed, and a decay factor removed, see equation [5].

Total stimulation calculation:

$$S_{total} = S_1 + S_2 - C_{FRS} * \text{DECAYRATE} \quad (5)$$
$$\text{if } S_{total} >= \text{MAXNEWCONC } S_{total} = \text{MAXNEWCONC} - 1$$

where:

S_{total} is stimulation from all $\mathbf{Ag}^f$s and masked $\mathbf{Ck}^f$s, reduced by decay
C_{FRS} is concentration of current **FRS**
DECAYRATE is decay rate constant (set to 0.1)
MAXNEWCONC is maximum concentration increase each iteration (set to 100)

This is then scaled to prevent excessive new stimulation each iteration and added to the current concentration of the **FRS**, equation [6]. If this concentration drops below MORTALITY, the **FRS** is removed from the network. Figure 5 describes the fractal immune network algorithm.

Concentration update calculation:

$$C_{FRS} = C_{FRS} + S_{total} * (\text{MAXNEWCONC} - S_{total}) / \text{MAXNEWCONC} \quad (6)$$

```
Iterate until halted:
  For each new data item

        map data item to fractal antigen $Ag^f$ according to eqn [1]

        present $Ag^f$ to the fractal immune network:
        does $Ag^f$ match an existing fractal recognition space FRS
        with difference less than ANTIGENMATCHINGTHRESHOLD?
        Yes:    calculate stimulation $S_1$ of best matching FRS according to eqn [3]
                *occasionally create new FRS by merging existing FRS shape with $Ag^f$
        No:     create new FRS with shape $Ag^f$

        for every FRS in the network:
                emit a fractal cytokine $Ck^f$ to every other FRS
        for every mature FRS in the network:
                mask all incoming $Ck^f$s with fractal receptor $Rc^f$
                does FRS match $Ck^f$ masked by $Rc^f$
                with difference less than CYTOKINEMATCHINGTHRESHOLD?
                Yes:    calculate stimulation $S_2$ of FRS according to eqn [4]

        increase age of every FRS, when age > MATUREAGE FRS is mature.

        update concentration of every FRS according to eqn [6]
        has concentration fallen below MORTALITY?
        Yes:    delete FRS
                is there more than zero FRSs in the network?
                No:     halt processing

  Every EVOLVEFREQ iterations, evolve mapping coefficients using eqn [2]and restart network.
```

Fig. 5. The Fractal Immune Network algorithm. Algorithm constants are set as follows: ANTIGENMATCHINGTHRESHOLD = 500, CYTOKINEMATCHINGTHRESHOLD = 500, MATUREAGE = 150, MORTALITY = 0.01, EVOLVEFREQ = 1500

5 Experiments

5.1 Experimental Motivation

From the beginning of this research it was clear that the mapping between data items and fractal antigens would be critical. For example, if all data items were mapped to almost identical fractal shapes, then the resulting network would be unable to distinguish between different classes of data. Contrast this with a good mapping from data to $\mathbf{Ag}^f$, which would be able to amplify even minor differences between data and enable useful clusters to form. Indeed, a good mapping would exploit the self-similarity of fractals, enabling (when desirable) data items that might look different to be correctly classified in the same class, and data items that might look similar to be correctly classified in different classes.

It was for these potential benefits that an evolutionary stage was incorporated into the fractal immune algorithm, enabling the data-to-**Ag**^{f} mapping to be subtly modified over time, guided by a fitness function.

Here we present preliminary experiments to demonstrate the evolution of this mapping and provide evidence to demonstrate that it improves clustering by the fractal immune algorithm.

This IRIS dataset was employed (a set comprising data items of four values, in three classes with 50 items in each class). The first 50 items were treated as "class 1" by the fitness function, with the remaining 100 being treated as "class 2." The system was set up as described in previous sections, with all constants set to their default values. Evolution was permitted to proceed for 2000 iterations, where each iteration comprised the formation of a new network and the random presentation of the data set ten times. The option to create new merged "mutant" **FRS**s was active during evolution.

5.2 Results

Figure 6 illustrates a good result (judged by the fitness function) obtained by the fractal immune network using the default mapping coefficient values, before evolution. It should be clear that the network is malformed, with some areas excessively connected and other areas completely unconnected. Figure 7 shows the networks obtained after evolution, both with merging inactive, and with merging active. It should be evident that both show "healthier-looking" networks, with two clear subnetworks for each class of data, and fewer unconnected **FRS**s.

Table 1 shows the classification performance as measured by the fitness function of the fractal immune network. Evolution has clearly found an improved mapping, which increases the ability of the network to classify the IRIS data, (whether the merging **FRS** option is active or not). Although the results are not perfect, at this stage we are focusing on the ability of evolution to fine-tune the data-to-**Ag**^{f} mapping. From findings in other work (Neal 2003), the networks are likely to provide better results when given more iterations to converge. With improved, perhaps non-linear mapping functions, it seems likely that evolution would be able to exploit the fractal proteins further and increase accuracy.

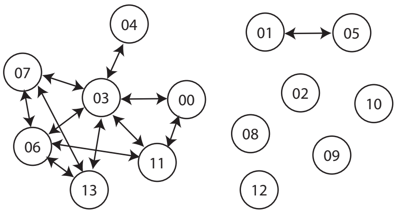

Fig. 6. Fractal Immune Network using default mapping coefficient values (merging inactive). **FRS**s are shown in arbitrary spatial positions.

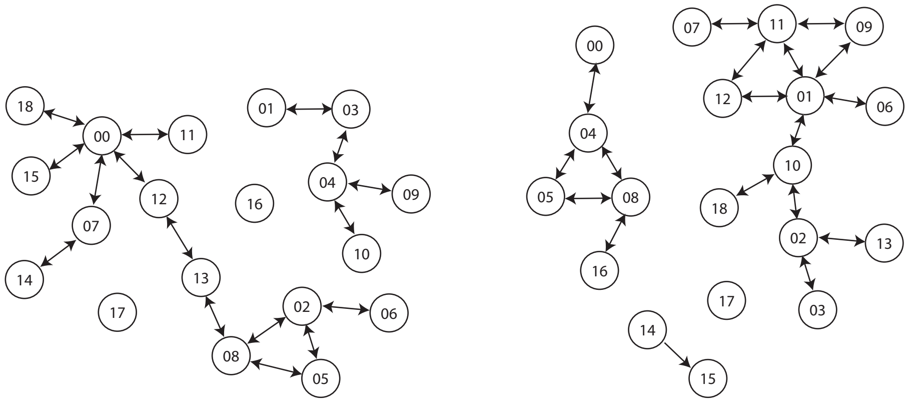

Fig. 7. Fractal Immune Network using evolved mapping coefficient values. Left: network obtained using no merging. Right: network obtained when new merged **FRSs** are permitted.

Table 1. Classification by the Fractal Immune Network. Note that **FRS**s within various subnetworks typically detect all data items, but only the two subnetworks that classify the largest amount of data in each class are shown here and used by the fitness function.

	Correct in class1	Correct in class 2	Incorrect in class 1	Incorrect in class 2
Default values	15 / 50	23 / 100	8 / 50	67 / 100
Evolved values, no merging	30 / 50	76 / 100	19 / 50	16 / 100
Evolved values, Merging	42 / 50	70 / 100	4 / 50	22 / 100

6 Conclusions

Proteins are the driving force in both development and immune systems. Here we have described the combination of fractal proteins (a model which has achieved great success within evolutionary developmental systems) with a network immune algorithm (which has also been demonstrated to great effect in the field of artificial immune systems). The model maps data items to fractal antigens, creates fractal recognition spaces in dynamic networks, and forms all network links by emission and reception of fractal cytokines. Experiments investigated the evolution of the mapping from data to antigens and demonstrated an improvement in cluster formation as measured by a fitness function. While not yet showing perfect results, these initial studies provide much promise. With a more advanced mapping stage and further analysis, the system provides the potential to enable the dynamics of this clusterer to evolve and thus automatically provide desirable clusters and data classification, regardless of the data.

Acknowledgements. Thanks to BAE Systems and Hector Figueiredo for providing support for this work. Thanks also to Johnny Kelsey for the idea of cytokines in the network algorithm.

References

1. Bäck, T. *Evolutionary Algorithms in Theory and Practice*. 1996. Oxford University Press, New York.
2. Bentley, P. J. Fractal Proteins. 2004. In Genetic Programming and Evolvable Machines Journal.
3. Bentley, P. J. Evolving Fractal Gene Regulatory Networks for Robot Control. 2003a. In Proceedings of ECAL 2003.
4. Bentley, P. J. Evolving Beyond Perfection: An Investigation of the Effects of Long-Term Evolution on Fractal Gene Regulatory Networks. 2003b. In Proc of *Information Processing in Cells and Tissues* (IPCAT 2003).
5. De Castro, L.N and Timmis, J (2002a). "An Artificial Immune Network for multi-modal optimisation". In proceedings of IEEE World Congress on Computational Intelligenece. Pp. 699-704.
6. De Castro, L.N and Timmis, J (2002b). "Artificial Immune Systems: A New Computational Intelligence Approach". Springer-Verlag.
7. de Castro, L. N. & Von Zuben, F. J. (2001), "aiNet: An Artificial Immune Network for Data Analysis", in *Data Mining: A Heuristic Approach*, H. A. Abbass, R. A. Sarker, and C. S. Newton (eds.), Idea Group Publishing, USA, Chapter XII, pp. 231-259.
8. Farmer, J. D., Packard, N. H. & Perelson, A. S. (1986), "The Immune System, Adaptation, and Machine Learning", *Physica 22D*, pp. 187-204.
9. Jerne, N. K. (1974), "Towards a Network Theory of the Immune System", *Ann. Immunol.* (Inst. Pasteur) 125C, pp. 373-389.
10. Kumar, S. and Bentley, P. J.. Computational Embryology: Past, Present and Future. 2003. Invited chapter in Ghosh and Tsutsui (Eds) Theory and Application of Evolutionary Computation: Recent Trends. Springer Verlag (UK).
11. Mandelbrot, B. *The Fractal Geometry of Nature.* 1982. W.H. Freeman & Company.
12. Neal, M. (2003), "Meta-stable Memory in an Artificial Immune Network", *LNCS 2787*. pp. 168-180. Timmis, J, Bentley, P and Hart E. (Eds). Springer-Verlag.
13. Perelson, A. S. (1989), "Immune Network Theory", *Imm. Rev.*, 110, pp. 5-36.
14. Perelson, A. S. & Oster, G. F. (1979), "Theoretical Studies of Clonal Selection: Minimal Antibody Repertoire Size and Reliability of Self-Nonself Discrimination", *J. theor.Biol.*, **81**, pp. 645-670.
15. Timmis, J. and Neal, M (2001). "A resource limited artificial immune system for data analysis" Knowledge Based Systems. 14(3-4).:121-130.
16. Wolpert, L., Rosa Beddington, Thomas Jessell, Peter Lawrence, Elliot Meyerowitz, Jim Smith. *Principles of Development, 2nd Ed.* 2001. Oxford University Press.