What have Gene Libraries done for AIS?♦

Steve Cayzer, Jim Smith[1], James A.R. Marshall[2],
Tim Kovacs[2]
Digital Media Systems Laboratory
HP Laboratories Bristol
HPL-2005-116
July 7, 2005*

AIS, immune systems, gene libraries, meta learning, Baldwin effect

Artificial Immune Systems (AIS) have been shown to be useful, practical and realisable approaches to real-world problems. Most AIS implementations are based around a canonical algorithm such as clonotypic learning, which we may think of as individual, lifetime learning. Yet a species also learns. *Gene libraries* are often thought of as a biological mechanism for generating combinatorial diversity of antibodies. However, they also bias the antibody creation process, so that they can be viewed as a way of guiding the lifetime learning mechanisms. Over time, the gene libraries in a species will evolve to an appropriate bias for the expected environment (based on species memory). Thus, gene libraries are a form of *meta-learning* which could be useful for AIS. Yet they are hardly ever used. In this paper we consider some of the possible benefits and implications of incorporating the evolution of gene libraries into AIS practice. We examine some of the issues that must be considered if the implementation is to be successful and beneficial.

* Internal Accession Date Only Approved for External Publication
♦ICARIS 2005 : 4th International Conference on Artificial Immune Systems, 14th-17th August 2005, Banff, Alberta, Canada
[1]Faculty of Computing Engineering and Mathematical Sciences, University of the West of England, Bristol, BS16 1QY, UK.
[2] Department of Computer Science, University of Bristol, Bristol, BS8 1UB, UK.

What have Gene Libraries done for AIS?

Steve Cayzer[1], Jim Smith[2], James A. R. Marshall[3] and Tim Kovacs[3]

[1] HP Laboratories, Bristol, BS34 8QZ, UK (SC)
[2] Faculty of Computing Engineering and Mathematical Sciences,
University of the West of England, Bristol, BS16 1QY, UK (JS)
[3] Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK (JM, TK)

Abstract. Artificial Immune Systems (AIS) have been shown to be useful, practical and realisable approaches to real-world problems. Most AIS implementations are based around a canonical algorithm such as clonotypic learning, which we may think of as individual, lifetime learning. Yet a species also learns. *Gene libraries* are often thought of as a biological mechanism for generating combinatorial diversity of antibodies. However, they also bias the antibody creation process, so that they can be viewed as a way of guiding the lifetime learning mechanisms. Over time, the gene libraries in a species will evolve to an appropriate bias for the expected environment (based on species memory). Thus gene libraries are a form of *meta-learning* which could be useful for AIS. Yet they are hardly ever used. In this paper we consider some of the possible benefits and implications of incorporating the evolution of gene libraries into AIS practice. We examine some of the issues that must be considered if the implementation is to be successful and beneficial.

1 Introduction

In any biologically inspired algorithm, one is obliged to make a number of concessions to simplicity. Indeed, one might argue that excessive biological realism is undesirable, since it will lead to building a system rather too specifically tailored to the biological environment. Nevertheless, the danger of oversimplification is that one may make a generic system, full of sweeping assumptions, that is not well suited for real world tasks [9]. In this paper we focus on two such broken assumptions. Firstly, *random creation of antibodies*. As any machine learning student knows, the naïve *generate and test* metaphor is the canonical algorithm, cheap yet unsystematic and often hopelessly inefficient. In the AIS world the approach may bring scalability problems [19]. The second broken assumption states that *antigens are uniformly distributed in non-self space*. We think this is unlikely to be representative of real world problems, and is certainly not true of the well-known UCI datasets [3].

In the biological system, of course, neither assumption holds. Firstly, antibodies are created from genes spliced from the so-called *gene libraries*; this ensures that antibody creation is far from random. Secondly, uniform coverage of non-self space is not only unnecessary, it is impractical; non-self space is too big! Thus, from a computational point of view, libraries introduce *initialisation bias* and provide a 'species memory' to tackle the antigen mapping task.

What could this mean for AIS? Could gene libraries be used to intelligently seed our algorithm? In the paper we consider whether gene libraries might:

1. improve non-self space coverage – through better placement of detectors (antibodies), over and above random creation;
2. reduce the cost of detector generation by more effectively avoiding self;
3. map the antigen population more accurately; and
4. help deal with co-evolving antigens

At a trivial level, the answer to all these questions is affirmative. Yet, of course, the computational cost of maintaining and evolving gene libraries may make the approach infeasible. In this paper, we outline a method for a principled evaluation of each feature. We include some preliminary results suggesting that option 2 is somewhat easier to achieve than option 1. Our work is intended to shed light on the sort of real world problems for which gene libraries should be considered.

We start by reviewing the relevant biology. We then consider the criteria outlined above in more detail before presenting our preliminary results. The considerable body of related work is reviewed before we make our concluding remarks.

2 Biological Metaphor

In this section, we give sufficient biological background for the purposes of this paper. This is not intended to be an exhaustive review; the interested reader is directed to sources such as Kuby [11] or Travers [16].

In the human immune system (HIS), gene libraries are used to generate both T cell (T cell receptor; TCR) and B cell (antibody) diversity. Antibody molecules are composed of four *immunoglobulin* chains; two identical pairs of heavy and light chains. Each chain contains a variable region which determines its antigen specificity; the DNA encoding this region is constructed by sampling from so-called V, D and J *gene libraries* (see Table 1); usually one from each, although sometimes multiple D segments can be sampled [16]. This DNA mixing occurs during B cell maturation, during which further diversity is encouraged by *junctional flexibility*, *P-additions* and *N-additions* (insertion or deletion of base pairs between gene library segments). Of course, such variability inevitably means many such generated gene sections are unviable. There are two interesting mechanisms to counter this. Firstly, there are two 'flavours' of light chain, κ and λ. If a viable κ gene segment cannot be built, then an attempt is made to build λ (thus typically κ is more prevalent than λ). Secondly, being diploid, B cells have 2 chromosomes for each immunoglobulin chain type. This allows two attempts to generate valid chain DNA (although a successful combination suppresses further attempts; this is *allelic exclusion*). Nevertheless, only about 10% of the pre-B cells in the bone marrow progress to maturity. Once a B cell is mature, however, it is immunologically committed[1].

The gene library mechanism appears at first to be wasteful: to make an immunoglobulin variable region of 223 amino acids we supply enough DNA to

[1] In fact there is a further choice to be made, at *transcription* (DNA to mRNA) time, about which C region to choose. However this *isotype switching* mechanism does not affect the antigen specificity, so it is not considered here.

encode 11,975 amino acids. However this redundancy enables 2M combinations (which if stored linearly would require 2M × 223 = 446M). Combinatorial diversity is further enhanced by the join errors described above, and by *somatic hypermutation*, in which B cells activated by antigens are stimulated to divide with a mutation rate of about 0.001 per base pair per generation (this compares to a spontaneous mutation rate of about 10^{-8}). To encode 223 amino acids needs 669 base pairs, so this mutation rate is expected to change one base pair almost every division. Estimates of total antibody diversity vary from 10^{10} [11] to 10^{14} [16], with somatic hypermutation pushing the number even higher to maybe 10^{16} [7]. The expressed diversity is, of course, likely to be somewhat lower because not all combinations are equally likely (in the mouse, some are actually disallowed). One assumes also that some variants are never expressed because they are autoreactive. Finally, it should be noted that only a subset (estimated 10^6-10^7) of these types are actually represented at any one time.

TCR diversity is similar, although a larger number of J segments (61) leads to a much higher gene library diversity, 10^{18} [16]. TCRs also do not undergo somatic hypermutation (perhaps a protection against generating autoreactive T cells). So for TCRs the diversity is more heavily germline encoded; conversely, some non-human species rely much more on somatic mechanisms.

Library	**Gene length (amino acids encoded)**		**Gene library size (number of gene segments)**			**Total**
	Heavy	Light	Heavy	Light κ	Light λ	
V	94	97	51	40	31	
D	3	N/a	27	0	0	
J	16	13	6	5	4	
(amino acids)	113	110	4971	3945	3059	**11,975**
(combinations)			8262	200	120	**2,643,840**

Table 1. Human antibody diversity generation. These numbers are taken from a single individual and are not the same for all individuals. The first three rows show the size of each V, D and J gene, and the number of genes in each library. It is a simple matter to sum the gene lengths to arrive at the total number of amino acids in an immunoglobulin chain variable region (4^{th} row, first 2 columns). Multiplying the library size and gene length shows the number of amino acids encoded in each gene library (e.g. 4971 = 51×94 + 27×3 + 6×16). Combining the gene library sizes (final row) shows the diversity generated from each library (e.g. 8262=51×27×6). Finally, using both light chain alternatives in combination gives the expected total diversity (1982880 = (8262×200) + (8262×120)). Note that since 3 base pairs encode one amino acid, you should multiply by 3 to get the number of base pairs. Adapted from Kuby [11].

However, even this extraordinary diversity may not be sufficient for all possible antigen encounters. In principle, since there are 20 amino acids, there are $20^{223} = 10^{390}$ ways of expressing the variable regions of an antibody. This renders almost negligible the antibody diversity expressible by an individual, even if we multiply this by every human on the planet. Bakács et al [28] concur, and point out that antigenic variation is a hallmark of several RNA viruses, thus providing a fast moving target. Yet we seem able to mount an immune response to pretty much any foreign molecule, "even those…never having appeared before in evolutionary time" [7]. One simplification is that vast swathes of this antibody shape space will be identical, or topologically infeasible, or simply non-functional. So it would seem reasonable that gene libraries bias the antibody creation process towards creating viable immunoglobulin chains.

But we suspect that some antigen shapes are simply more likely than others, due to the energetics of protein folding. Perhaps gene libraries may help to focus the antibody creation process into the most promising areas of shape space, a possibility suggested by the fact that multiple antibody types bind a single antigen [7], and the finding that V region genes are clustered into related families and clans [16].

Taking inspiration from this account, we can see that gene libraries, shaped by evolution, are used to guide the B cell creation process to create antibodies with a good chance of success, while preserving the ability to respond to novel threats. This has obvious parallels in AIS, in instances where random creation does not scale, or where memory enabled by lifetime learning mechanisms is not sufficiently persistent.

3 What Are Gene Libraries For?

3.1 Enhanced Coverage

The most naïve way of looking at antibody creation is a way of covering a multidimensional area (antigen space). If one uses gene libraries to bias the creation process, then it is easy to see that evolution should encourage the emergence of diverse gene libraries, which perform some coarse grain mapping on antigen space. Indeed, Oprea and Forrest [21] found precisely this mechanism at work. Yet in the real world, such a mechanism is highly expensive. If all one wants to do is to cover a well understood antigen space (say, binary strings with Hamming distance matching), then an enforced distribution would be the simplest mechanism. There are well understood algorithms for dealing with other types of spaces, for example Wierzchon's schema match [26] for r-contiguous matching.

Of course, generally the task is to map antigen space *while avoiding self*. This is a more involved task, yet even here there are simple algorithms that might do a superior job to gene libraries, particularly when computational cost is taken into account. For example, de Haeseleer's greedy algorithm [6] generates a number of non-self 'templates' and uses these to create an antibody which binds to the most unmatched antigens. Singh [23] extended this algorithm to deal with non-binary alphabets. A somewhat simpler approach is Ayara's NSMutation [2] which generates detectors randomly, mutating those that match self. Gonzalez [12] uses idiotypic suppression to maximise antibody diversity. Finally, Wierzchon's schema matching algorithm [26] can be used to effectively generate self-avoiding, non-self-matching antibodies [25].

This, then, provides a convenient place to start our investigation into gene library function. We propose a comparative study on these algorithms in order to find what characteristics of a real world problem (if any) would suggest the use of gene libraries might be advantageous. It should be noted that complete coverage will almost certainly be impossible in presence of self [6]. An advantage of using a binary string representation with r-contiguous bits is that one can work out the theoretical optimum [26] and compare the coverage obtained by any one individual against it.

3.2 Avoiding Self

As mentioned in the last section, simple coverage is probably not a sophisticated enough aim for gene libraries. The avoidance of self is a different slant on the same problem. In the HIS, this is essential to protect against autoimmune reactions. Whilst it is true that there is a negative selection mechanism operating in the HIS, it would clearly be beneficial for the creation process to have a bias *against* creating self reactive antibodies. In the context of AIS, this amounts to making the creation process cheaper. In other words, the number of attempts to create a valid (i.e. not self-reactive) antibody should be considerably *less* using gene libraries than a naïve random approach, which is exponential in the size of self [8]. Of course the benefit may still not justify the computational expense of using gene libraries. Also, alternative algorithms (see the last section) may be a rather cheaper way of attaining the same benefit. Thus, the cost of avoiding self is a plausible evaluation function which we can use as an additional comparison point.

3.3 Mapping Antigens

Antigens correlate to things we want to detect, or classify. In a sense, they are 'points of interest' in the non-self space. If we accept that it is impossible (given the computational resources, i.e. number of antibodies available) to map all of non-self space, even in principle, then it clearly behoves the system to bias antibody creation towards these areas of interest.

This, now, starts to move towards more realistic scenarios. Imagine, for example, a document classifier that identifies 'interesting' documents. Given a training set of interesting and uninteresting documents it will generate a set of detectors to identify, and generalise from, interesting documents. Such documents tend to form clusters in non-self space. A gene library would bias the creation process so that rather than fumbling blindly in non-self space, antibody creation would be guided towards the clusters of interest.

A more subtle point is that gene libraries provide a long term memory. Say, for example, that a set of randomly created antibodies are subject to clonotypic learning, such that they cover the clusters in one particular training set but not all clusters ever seen. Gene libraries provide a way of remembering past encounters so that antibody creation is more likely to match novel clusters which are nevertheless similar to those seen some time ago. This motivation has guided several previous implementations of the gene library metaphor [14,20]. In a variant of this approach, gene libraries have been used as metaphor in an email filtering system [31]. Here words found in "interesting" emails were archived, then during the mutation stage of clonal selection, a word (gene) chosen to be mutated was replaced by one chosen from the archive, rather than subjected to random perturbation.

Of course, evolving these gene libraries will take time, during which a great deal of random searching (or searching guided by a cheap heuristic) might have taken place. Whether the trade-off is worth it is a moot point, and will depend greatly on the problem characteristics. For example, there is a choice about whether to use self; if so, the task is transformed into a 3-class problem (self, non-self, don't care). Another

consideration is the evaluation function, for example: number of antigens successfully detected after X generations; number of antibodies created before all antigens detected; average cost before 1st antigen detected. Our work should provide some principled guidance for AIS practitioners.

3.4 Winning the Evolutionary Arms Race

A further consideration is that in the real world, antigens (i.e. whatever you want to map) are unlikely to stay still. Mapping dynamic antigens is a more complex problem, akin to non-stationary landscapes, for which problem generators [24] are available. The species-level memory afforded by gene libraries could prove to be a boon here, since they may evolve to track a moving target. This assumes that gene libraries can 'keep up' with the rate of antigen change, of course; a point illustrated by Oprea and Forrest's work [21].

A further complication is that antigens may be moving purposefully, to avoid detection by the immune system. For example, a sensible strategy for an antigen is to get close to self, ideally within a 'cove' or 'hole' [6]. Such possibilities can be explored using coevolutionary models, and this is a major focus of our ongoing research. Interestingly, Gathercole and Ross [10] use a predator-prey model for effective coverage, which links in nicely with our original scenario.

What practical implications could such work have? In almost all real world problems the target is constantly moving (for example, the definition of: an 'interesting' or 'relevant' document; an anomalous network event; a suspected fraudulent mortgage application, a spam email). Could gene libraries provide the basis for a more robust, adaptive and responsive system?

4 Implementing Gene Libraries

The preceding discussion describes the areas where we intend to focus our investigation. As a validation scenario, we repeated Forrest's canonical experiments on self/non-self discrimination [8]. In this paper, the authors use a random generate and test algorithm to create a set of detectors covering non-self with some desired (low) probability of failure. They used a simple binary string universe with self, non-self and an r-contiguous matching measure. For example, using a 32 bit string with 16 self strings and r set to 8, 105 attempts were needed to create 46 detectors which between them covered about 90% of non-self space.

It is worth commenting a little on this experimental setup. The r-contiguous matching function is often argued to be more biologically plausible than Hamming distance, although this view is strongly criticised by Timmis & Freitas [9]. In fact, Forrest et al., who pioneered the metric, call it "[an] arbitrary decision [made] in order to simplify the mathematical analysis". As an example, they calculate the chance of 2 random strings matching as roughly 0.05 using the parameters above.

In an elegant paper, Wierzchon [26] extended this mathematical analysis to sets of detectors. In general, non-self space is likely to contain 'holes' that cannot be detected [6]; these are regions where matching detectors would also match self. Wierzchon

showed how to calculate the number of holes for any given self set, giving a upper bound to the amount of non-self coverage possible. In a similar vein Esponda, Forrest and Helman have analysed the trade-offs between postivie and negative selection for the r-chunks matching function [29], arguing that this is a prefrable matching function, and Stibor, Bayarou and Eckert have investigated the propoerties of this matching function as the underlying alphabet is extended beyond the binary case [30].

The evaluation measure is also important. In fact, we would say it is key to understanding the system. The evaluation function in some sense defines the task. One measure is the ease of avoiding self, or equivalently the cost of generating a system. The detection rate, in contrast, measures the performance of the generated system. It can be calculated either for a fixed number of detectors (number of antigens matched divided by number of antibodies in system), or for a fixed number of attempts to generate a detector.

Following Forrest then, we used a set of antigens (sampled from a fixed population). The antibodies are subject to negative selection, but in our case rather than employing random creation we create the antibodies from gene libraries. The overall algorithm is as follows, where `TERMINATION_CRITERION` occurs when NR non-self antibodies have been generated, or the full set of combinations has been tested, whichever is sooner.

```
Create self
Create non-self
LOOP foreach generation
   LOOP foreach individual (= gene library)
      While (TERMINATION_CRITERION not met)
      DO
         Choose genes from gene libraries
         Create antibody
         IF match self THEN destroy
      END WHILE
      Evaluate fitness of individual
   END LOOP
   Do selection/recombination/mutation of individuals
   Do replacement of individuals
END LOOP
```

4.1 Some Preliminary Results

We conducted a series of experiments looking at the effort required (number of antibodies generated) to produce a set of `NR` (= 46) detectors, and the coverage they provide of the non-self region, as defined by its complementarity to a randomly created set of 128 “self” strings. The parameters are shown in table 2.

Initially we considered random generation to verify that our system was equivalent to Forrest’s. For 1000 repetitions we generated strings at random until we had created a set of 1024 non-self strings (i.e., not matching the self region), noting the total number of strings created (`NS_Attempts`). We then generated antibodies at random, discarding those which matched self until we had `NR` detectors, again noting

the effort required (number of antibodies created, NR0). Finally we measured the coverage provided by the set of detectors, calculated as the percentage of the non-self set which were matched by at least one of the detectors. These results are shown in Table 3. The coverage values obtained are higher than those Forrest noted, but their measure was slightly different – they changed one eight-bit block of one string from self and saw whether any of the detectors matched the mutated string. The values for NR0 are similar – they reported a mean of 34,915 with a standard deviation of 8513.

Parameter	Choice	Comments
Antigen representation	Binary string	32 bits
Antibody representation	Binary string	32 bits
Matching function	r-contiguous	8 bits
Antibody creation	Try a maximum of LC times from gene library or until NR antibodies created	LC =8000; NR =46 No mutation at this stage.
Fitness evaluation	% antigens matched from f antigens sampled randomly from non-self space.	f = 1024. Static; not changed between individuals or generations
Self	S randomly created strings.	s = 128. Static; not changed between individuals or generations
Genotype	20V11, 20D10, 20J11 LC = 20×20×20 = 8000	20V11: V library has 20 genes each with 11 bits.
GA parameters	Mutation: 0.01 per bit Crossover: 0.1 one point Selection: Binary Tournament Replacement: generational, no elitism Population size: 128 Learning: 500 generations	Tournament selection Avoids many problems with fitness-proportionate selection
Fitness of an individual	See below	

Table 2. Parameters and choices for our experiments

Measure	Min	Max	Mean	Std.Dev	Skewness
NS_Attempts	433,846	1,543,945	777,525.3	147027	0.718
NR0	16,863	80,105	35135.37	8386.91	0.803
Coverage	96.68	99.8	98.67	0.46	-0.61

Table 3. Results of Random Antibody Generation.

Next, we considered the effects of evolving gene libraries. During the evolution, each population member was evaluated as follows. First, a random permutation of the LC possible antibodies was generated. The antibodies were then created and tested for matches against self in this order, until either the number of detectors created (ND) equalled NR, or all possibilities had been exhausted (NR0 = LC). In addition to ND and NR0, the efficiency and coverage were calculated and the mean per generation noted, defined as:

$$Efficiency = \frac{ND}{NR} * \frac{(LC - NR0)}{(LC - NR)},\ Coverage = \frac{match(NonSelf)}{all(NonSelf)} * 100\%$$

We considered three different fitness functions to be maximised: efficiency, coverage, and a equal linear combination of the two. Preliminary experiments used a simpler version of efficiency, which discarded individuals which could not produce

the full set of NR detectors. This proved to be highly ineffective since in the early generations many population members had ND < NR and so were assigned fitness 0.0, removing any fitness gradients and preventing evolution. For each fitness function we conducted 25 runs, each using a different set of self and non-self, but the same set of seeds were used across the fitness functions to avoid the possibility of one seeing "easy" or "hard" sets of self/ non-self strings.

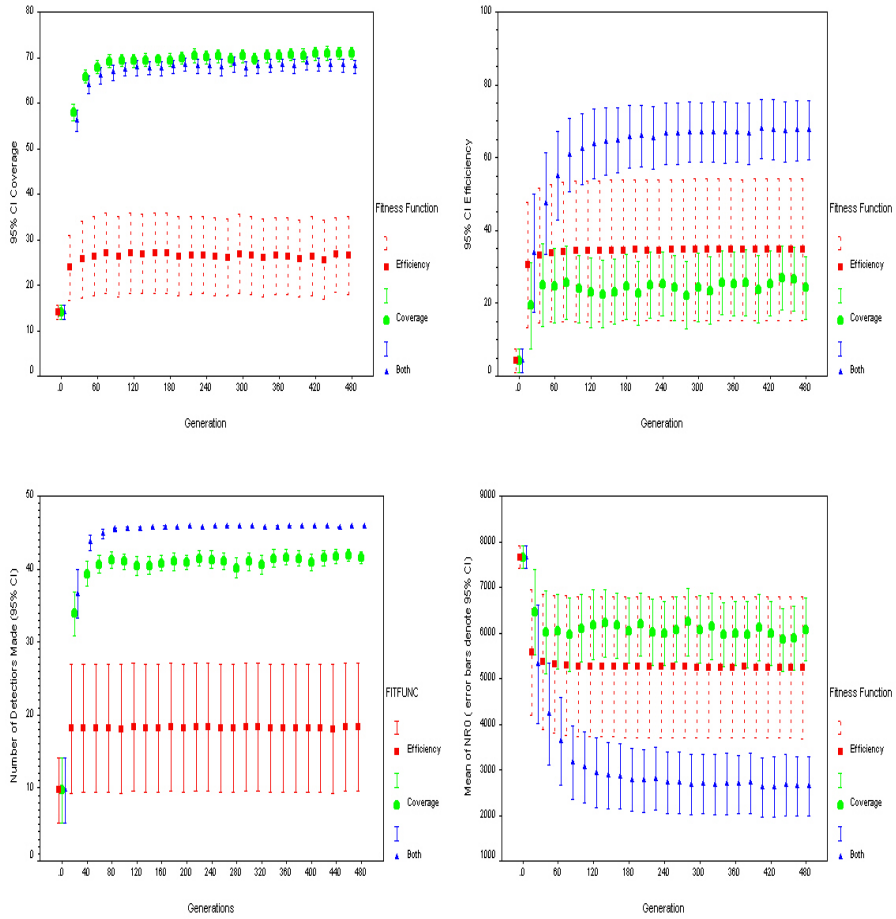

Fig. 1. Evolution of metrics under different fitness functions: Coverage (top left), Efficiency (top right), Number of detectors made: ND (Bottom left), Effort: NR0 (bottom right). Markers are for mean over 25 runs, with error bars showing 95% confidence interval for mean.

Figure 1 shows the evolution of the metrics considered under the three different fitness functions. From this figure we can make the following observations:

- Evolution solely under the influence of avoiding self (efficiency) produces results which are highly variable and are, on average poor. Closer examination shows that this is because around half of the runs cannot produce solutions providing many, let alone the full complement of, detectors.

- Evolution under the influence solely of coverage of non-self, reaches around 70% coverage, but produces less detectors (ND ~40) and uses considerable effort to produce them (NR0 ~6000)
- Evolution under the combined influence of efficiency and coverage quickly learns to produce NR detectors at low cost (NR0 < 3000), while reaching similar levels of coverage (maximum recorded 77.2%, mean over last 400 generations 68.03% with std. deviation 3.06). Clearly in this case the GA is managing to evolve gene libraries which efficiently avoid self and cover the non-self regions.

In each case evolution has nearly stopped by the 100^{th} generation. Taking the results for the combined fitness function, and sampling every tenth of the last four hundred generations gives us 25 runs × 40 samples = 1000 samples to compare with the random results. Statistical analysis using a variety of non-parametric tests confirms that the coverage is worse than the random case (68% vs 98%), but that the effort required to avoid self is considerably (tenfold) lower.

4.2 Evaluation and Extensions

It would appear that the use of evolved gene libraries can produce significant improvements in terms of the efficient generation of antibodies which do not match self, but that the resultant coverage of the non-self regions is clearly not uniform. One possibility is that the gene libraries are in fact modelling self, or more accurately its inverse (in the r-contiguous matching sense). Another is that during the initial phase of evolution fitness gains are made by improving efficiency, but that diversity is lost, so that coverage does not improve as much. These will be tested in future work by incorporating explicit diversity preservation measures into the genetic algorithm.

To investigate whether gene libraries could produce reasonable coverage, we simplified the gene libraries so that there was only one library (46V32). We were now able to obtain the 90% coverage reported by Forrest et al. In addition, if we allow the AIS a fixed number of attempts (NR0 =105) to generate as many detectors as it can, then the performance goes to about 95% with about 60 detectors. This, of course, amounts to using a GA to search for a set of optimal antibodies, which is rather less ambitious than a search for a set of optimal gene libraries. What is rather interesting is that similar results were obtained even without using any self. This adds weight to our supposition that the block for gene libraries is the capability of covering a large space with a small number of detectors, rather than the problem of avoiding self.

It is worth emphasising that improved results could probably be obtained by using a more sophisticated GA, or one with a larger population. We could also combine the fitness measures together using a Pareto-based approach [1]. Our parameter choices were domain led - we chose to use one-point crossover because of its positional bias: it is more likely to keep together adjacent genes than, for example, uniform crossover. The gene library fragments are coded as contiguous segments.

One problem of fitness evaluation is that there are a large number of potential antibodies to be evaluated before we get a good idea of the ‘worth’ of a gene library. Other authors have faced the same problem, and either limited evolutionary learning to a partial fitness measure [15] or resorted to less computationally expensive

algorithms, such as hill climbing [21]. Also, the 'meaning' of a gene in a library depends on which genes it is combined with (a form of *epistasis*).

An important consideration here is the use of gene libraries for coding efficiency. As discussed above, the HIS encodes 2M combinations of 223 amino acids using 12000 amino acid coding units[2], a compression rate of 223×2M:12K = 37K:1. Our naïve gene libraries managed to encode 8000 combinations of 32 bits using 640 bits, a compression ratio of only 400:1. Clearly, there is trade-off between encoding efficiency (the most efficient being a random generator) and preservation of species memory (the extreme being 1 gene per antibody, as per our 46V32 encoding).

We made several simplifications to the biology, some of which are straightforward to explain. For example, no lifetime learning mechanism is included here; we are simply testing for the initial coverage. Further work is likely to involve a mechanism for clonotypic learning; this would be a good way to introduce a Baldwin effect which may well be essential to leverage gene library diversity and hence achieve better coverage [21]. Another example is the encoding, which was binary, rather than, say, 20 letters (for amino acids) or 4 letters (for DNA bases). The motivation here is to evaluate gene libraries in established, analytically tractable, scenarios.

Other possibilities suggest interesting areas to explore. For example, we used a static definition of self, using random selection of 128 binary strings. Different sizes of self may influence the results, as might the distribution of self (should it be clustered?). Self might also be dynamic, though in a principled way. Perhaps inter-individual variation could be captured using Gaussian perturbation, while species drift could be modelled by including self in an individual's genotype. The antigens used here consisted of 1024 randomly created strings (screened against self): an enforced uniform selection might be a more principled way to check coverage. Another possibility is to use a static, perhaps clustered definition of antigen, which could be generated from its own set of gene libraries or antigen schema. Such possibilities of course take us into the more complex scenarios introduced earlier.

5 Related Work

We make no claim to be the first researchers to look at gene libraries. That claim, at least for AIS, perhaps belongs to Stephanie Forrest's group at the University of New Mexico. Perelson et al [22] showed that gene libraries can enhance coverage. They constructed antibodies by taking segments from 4 gene libraries. Fitness was evaluated by calculating the coverage of a varying size antigen universe. Unsurprisingly, the fitness asymptotically approached random levels as the antigen universe size increased, but approached perfection as the number of antigens decreased. Their work differs from that presented here in that the matching function used was Hamming, and (more significantly) there was no 'self' to avoid. Interestingly, they experimented with both straight Hamming scores (fitness of an antibody is its closeness of match with an antigen) and thresholds (fitness of an individual is the number of antigens to which at least one antibody binds sufficiently).

[2] Recall an amino acid coding unit is 3 base pairs of DNA

They also show that a modest degree of somatic (clonotypic) learning may provide a Baldwinian acceleration.

Extending this work, Hightower et al [15] investigated the evolution of effective coverage. They showed that the 'best' coverage was achieved by a high Hamming distance (spread out antibodies) – but not too high. A maximal separation actually allows gaps in coverage (analogous to gaps between disjoint spheres). Oprea & Forrest [21] showed that as the pathogen set size decreases, the structure of the gene library changes, moving from a 'coarse mapping' of antigen space towards a more focused targeting of pathogenic clusters. They also show that since gene library size increases coverage only logarithmically, it must be augmented with somatic learning.

Forrest et al [7] do not use explicit gene libraries, but they do consider bias in the antibody creation process, which they evolve to map an antigen population. Interestingly, this process requires a minimal number of antibody types to bind to a particular antigen.

Other groups have also studied gene libraries. Hart & Ross [13] used a genetic algorithm (GA) to evolve libraries for a scheduling immune system. Essentially, the gene libraries preserved useful fragments of antibody (building blocks) that could successfully be reused. The same authors [14] develop this notion by suggesting that the germline (gene library) could be 'seeded' with antibodies during learning. Coello Coello et al employ a similar approach [27]. We would suggest that such an approach is akin to having species level memory cells. Kim & Bentley [17] mention the notion of gene libraries as a way of encoding 'some knowledge of antigens'. In their companion paper [18] they model gene libraries as a single population of successful genes which are combined to form detectors. Thus the gene library evolves in parallel with AIS itself. More recently, these authors [20] have used deleted memory detectors as 'gene library' – i.e. long term memory. Kim and Bentley point out that "the fact they managed to become memory detectors at all implies that they hold valid information about non-self antigens in previous clusters".

In much of this work [13,14,17,18,20] the gene library metaphor is used as an engineering artefact rather than in the more biologically faithful way that we investigate here. Conversely, the theoretical work [7,15,21,22] is distanced from existing AIS applications (e.g. by absence of self). The current work attempts to build a bridge between the established theoretical foundations and current AIS engineering practice.

6 Conclusion and Future Directions

In this paper, we have shown that gene libraries are an interesting, and perhaps useful, tool in the AIS practitioner's repertoire. We have outlined some areas where gene libraries might help and shown how to evaluate gene libraries in each area. Our preliminary results suggest that gene libraries may not be well suited to simply enhancing coverage, and may be better employed for improving the average quality of created antibodies (here exemplified by the task of avoiding self). These results need to be extended and analysed, and we have proposed a plan for testing the gene library metaphor in progressively more complex scenarios.

References

1. Anchor, K.P., Zydallis, J.B., Gunsch G.H.,Lamont, G.B.: Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach. In: J. Timmis and P. J. Bentley, (eds.) Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS) Canterbury, UK. (2002) 12-21
2. Ayara, M., Timmis, J., de Lemos, R., de Castro, L.N., Duncan, R.: Negative Selection: How to Generate Detectors. In: J. Timmis and P.J. Bentley (Eds.) 1st International Conference on Artificial Immune Systems, University of Kent at Canterbury, September 2002. (2002) 89-98
3. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science. (1998) Accessed 25 April 2005.
4. Burnet, F.M. (1959) The clonal selection theory of immunity. Nashville, TN: Vanderbilt University Press
5. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Approach. Springer-Verlag, London UK (2002)
6. D'haeseleer P., Forrest, S., Helman, P.: An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. In: Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy (1996)
7. Forrest, S., Javornik, B., Smith, R.E., Perelson, A. S.: Using genetic algorithms to explore pattern recognition in the immune system. Evolutionary Computation, Vol. 1, No. 3 (1993), 191-211.
8. Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA: IEEE Computer Society Press. (1994) 202-212
9. Freitas, A., Timmis, J.: Revisiting the Foundations of Artificial Immune Systems: A Problem Oriented Perspective. In: Timmis, J., Bentley, P., Hart, E. (Eds.) Proceedings of the 2nd International Conference on Artificial Immune Systems. Lecture Notes in Computer Science Vol 2787 Springer-Verlag, Berlin Heidelberg New York (2003) 229-241
10. Gathercole, C. Ross, P.: Dynamic Training Subset Selection for Supervised Learning in Genetic Programming. In: Davidor, Schwefel and Manner (eds.): PPSN III: Proceedings of the 3rd Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, Vol. 866. Springer-Verlag, Berlin Heidelberg New York (1994) 312—321
11. Goldsby, R.A., Kindt, T.J., Osborne, B.A., Kuby, J.: "Immunology" W H Freeman, New York 5th edition (2003)
12. González, F., Dasgupta, D.: Combining Negative Selection and Classification Techniques for Anomaly Detection. In: Proceedings of the Congress on Evolutionary Computation, Honolulu, Hawaii May (2002) 705-710
13. Hart, E., Ross, P.: An Immune System Approach to Scheduling in Changing Environments. In: W.Banzhaf, J.Daida, A.E.Eiben, M.H.Garzon, V.Honavar, M.Jakiela, R.E.Smith (Eds.) Proceedings of Genetic and Evolutionary Computation Conference (GECCO) July 13-17 Morgan Kaufmann (1999) 1559-1566
14. Hart, E., Ross, P.: The Evolution and Analysis of a Potential Antibody Library for Job-Shop Scheduling. In: New Ideas in Optimisation. D. Corne, M.Dorigo & F. Glover (eds), K. McGraw-Hill, London. (1999) 185-202
15. Hightower, R., Forrest, S., Perelson, A.S.: The evolution of emergent organization in immune system gene libraries. In: Proceedings of the Sixth International Conference on Genetic Algorithms, Los Altos, CA. Eshelman L.J. (Ed.) Morgan-Kauffman, San Francisco, CA (1995) 344-350

16.Janeway, C.A., Travers, P., Walport, M., Shlomchik, M.: "Immunobiology: The immune systems in health and disease" Garland Publishing, New York. 5th edition (2001)
17.Kim, J., Bentley, P., The Human Immune System and Network Intrusion Detection. In: 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99), Aachen, Germany, September 13- 19. (1999)
18.Kim, J., Bentley, P.: The Artificial Immune Model for Network Intrusion Detection. In: 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany, September 13- 19. (1999)
19.Kim, J. and Bentley, P. J.: Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection. In: Genetic and Evolutionary Computation Conference 2001 (GECCO-2001), San Francisco, July 7-11, 2001. (2001) 1330 - 1337
20.Kim, J., Bentley, P. J.: A Model of Gene Library Evolution in the Dynamic Clonal Selection Algorithm. In: Proceedings of the First International Conference on Artificial Immune Systems (ICARIS) Canterbury, pp., September 9-11, 2002. (2002) 175-182
21.Oprea, M., Forrest, S.: Simulated evolution of antibody libraries under pathogen selection. In: Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA (1998)
22.Perelson, A.S., Hightower, R., Forrest, S. "Evolution and somatic learning in V-region genes. Research in Immunology 147 (1996) 202-208
23.Singh S.: Anomaly detection using negative selection based on the r-contiguous matching rule. In: J. Timmis and P. J. Bentley, (eds.) Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS) Canterbury, UK. (2002) 99-106
24.Spears.W. (contact) Repository of Test Problem Generators. Available at: http://evonet.lri.fr/evoweb/resources/software/record.php?id=393 Accessed 22 April 2005
25.Wierzchon, S.T.: Generating Optimal Repertoire of Antibody Strings in an Artificial Immune System. In M. Klopotek, M. Michalewicz and S. T. Wierzchon (eds.) Intelligent Information Systems. Advances in Soft Computing Series of Physica-Verlag/Springer Verlag, Heidelberg/New York, Physica-Verlag. (2000) 119-133
26.Wierzchon, S. Deriving a concise description of non-self patterns in an artificial immune system. In: New Learning Paradigms in Soft Computing Physica-Verlag (2002) 438-458
27.Coello Coello, C. A., Rivera, D. C., Cortés, N. C.: Use of an Artificial Immune System for Job Shop Scheduling. In: J. Timmis, P. J. Bentley, E. Hart (eds.) Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS) Edinburgh, UK. (2003) 1 – 10
28.Bakács, T., Szabados, T., Varga, L., Tusnády, G.: Axioms of mathematical immunology. Studia Scientiarum Mathematicarum Hungarica 38 (2001) 13-43
29.Esponda, F., Forrest, S., Helman, P.: A formal framework for positive and negative detection schemes. IEEE Transactions on Systems, Man and Cybernetics Part B 34 (2004) 357–373
30.Stibor, T., Bayarou, K., Eckert, C.: An investigation of R-chunk detector generation on higher alphabets. In: LNCS 3102. (2004) 26–30
31.Secker A and Freitas A and Timmis J (2003). AISEC: An Artificial Immune System for E-mail Classification. Proc. Congress on Evolutionary Computation pp 131-139. IEEE