An Immune System Approach to Scheduling in Changing Environments

Emma Hart, Peter Ross
Division of Informatics
The University of Edinburgh
Edinburgh, EH1 1HN, Scotland
{emmah,peter}@dai.ed.ac.uk

Abstract

This paper describes the application of an artificial immune system, (AIS), model to a scheduling application, in which sudden changes in the scheduling environment require the rapid production of new schedules. The model operates in two phases: In the first phase of the system, the immune system analogy, in conjunction with a genetic algorithm, (GA), is used to detect common patterns amongst scheduling sequences frequently used by a factory. In phase II, some of the combinatoric features of the natural immune system are modelled in order to use the detected patterns to produce new schedules, either from scratch or starting from a partially completed schedule. The results are compared to those calculated using an exhaustive search procedure to generate patterns. The AIS/GA analogy appears to be extremely promising, in that schedules corresponding to situations previously encountered can easily be reconstructed, and also in that the patterns are shown to incorporate sufficient information to potentially construct schedules for previously unencountered situations.

1 Introduction

In the context of information processing, the natural immune system can be regarded as a highly parallel intelligent system, and yet has received relatively little attention compared to genetic algorithms and neural networks. In particular, its ability to perform pattern recognition tasks, to memorise patterns that have been seen previously, and to use combinatorics to construct efficient pattern detectors make it particularly attractive as a model for solving many real-world based problems. Some of the seminal work in this emerging field is described in (Dasgupta, 1998). This paper describes some of the key features of the way in which the biological immune system functions, and then illustrates how several parallels can be drawn between it and a real world scheduling problem, suggesting a possible method of tackling the real-world problem using techniques inspired by the natural immune system.

The biological immune system defends the body against invading pathogens, (antigens), that may be harmful by producing antibodies which recognise and remove the invaders. Remarkably, despite the fact that the body has fairly limited genetic resources, and there are an almost infinite number of possible pathogens, the immune system is able to react rapidly and efficiently to both those antigens it has previously encountered, as well as entirely new ones. The immune system is responsible for continually monitoring the body for unexpected changes, and then reacting in the appropriate manner to counteract any ill-effects, making use of both its long term memory abilities and its capacity to generate new antibodies.

Now consider a manufacturing scheduling environment:- while a factory is running, many environmental changes occur which will result in modifications having to be made to the factory scheduling system in order to keep things running smoothly. The nature of these changes is varied and covers an almost unlimited range of possibilities. Some events occur frequently and are predictable, whilst others are completely unpredictable. An efficient scheduling system should be able to react quickly to such changes, and either alter the current schedule or rapidly produce a new schedule to take such changes into account, so that manufacturing continues in an efficient manner. Hence, a direct analogy can be drawn between the two systems.

Although there have been numerous studies of the application of genetic algorithms, (and other optimisation techniques) to the scheduling domain, (Bäck et al., 1997), almost all have concentrated on producing a single schedule that is as close to optimal as possible, in the sense that one or more objectives are minimised. In the real-world however, such ideal schedules often cannot be directly implemented if variations in the environment in which the schedule was produced for have occurred. Also, 'optimal' schedules are often extremely fragile to slight perturbations in conditions, which result in them being rendered useless by even slight changes.

Analysis of data relating to scheduling problems supplied by real companies reveals that although deviations in the scheduling process often occur, the situations leading to the deviations are often predictable and there are generally known methodologies for dealing with them. An experienced scheduler can quickly piece together a new schedule using prior knowledge from past experiences, and it is rare to have to completely redesign a schedule. For instance, a number of typical scenarios can be imagined; certain machines commonly break down or require maintenance, or parts required to complete some jobs often arrive late from other parts of the factory. In such situations, a scheduler often acquires some kind of knowledge of how to cope with such situations by judicious scheduling of other jobs, leading to good schedules.

It appears reasonable that in many real-life situations a set of historical complete or partial schedules is available for use when trying to reschedule. Careful examination of this set of schedules reveals that various patterns commonly occur in subsets of the schedules. For instance, simply considering the order in which jobs are processed on machines it becomes obvious that there are particular groups of jobs (of varying size) that tend to occur in close proximity or in some particular order in more than one schedule. For example, common job sequences may be observed, such as "operation always occurs directly before operation b", or "operations a,b,c tend to occur in a group in many schedules, but in different permutations".

Thus, if a set of common patterns or parts of schedules could be built up using the knowledge encapsulated in past schedules, then these patterns can be used as 'building blocks' when constructing a new schedule.

The 'building block' idea is also observed in the real immune system, which builds antibodies to attack invaders from a germ-line or 'store' of DNA building blocks. The mammalian immune system has evolved over thousands of generations during which time it has been exposed to a diverse variety of pathogens in such a way that the germ-line contains a small, but highly efficient set of DNA blocks that can be used to generate antibodies that recognise almost all known pathogens. In a similar manner, in order to apply the analogy to our scheduling environment, we wish to evolve a store of schedule building blocks, that capture information acquired through past experiences, and that can be used to efficiently and rapidly construct new schedules.

This paper describes a system we have designed called *PRAIS*, (**P**attern **R**ecognising **A**rtificial **I**mmune **S**ystem), that is used to construct schedules using a simple scheduling scenario — j jobs, each with different arrival dates, due-dates and processing times, must be scheduled on a single machine, m. The ideal arrival pattern, and hence optimal machine sequence is supposedly fixed, however deviations in arrival dates occur, resulting in alterations having to be made. The job-sequence on the machine is an obvious candidate for observing patterns, and hence this attribute is used in this paper. Note however that there are other attributes of the system in which patterns may be observed, for example distribution of idle times on machines.

2 An AIS Model for Pattern Recognition in Scheduling

Previous work by (Forrest et al., 1993), and (Smith et al., 1993), has shown that an immune system model combined with a genetic algorithm can be used to evolve a set of antibodies that recognise a range of diverse, binary antigen strings. This work showed that an immune system model could both detect common patterns (schemas in the binary case) in a noisy environment and also maintain diversity in that many types of antibody evolved in niches, each niche responsible for recognising a particular antigen. Moreover, they showed that it was possible to control the evolution of the antibodies to be either 'specialist' (i.e the antibody only recognised a single specific antigen), or 'generalist' (i.e the antibody recognised a wide range of antigens) by varying the parameters of the genetic algorithm.

Forrest's system contains almost all the features that need to be realised in the scheduling domain, if we consider an *antigen* to represent a sequence of jobs on a particular machine, given a particular scenario, and an *antibody* to represent a short sequence of jobs that is common to more than one schedule. However, some modifications are required:- the proposed sequence-recognising AIS should produce antibodies

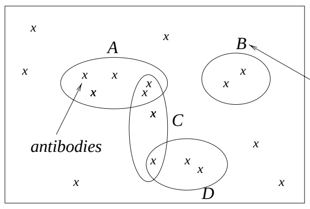

Figure 1: A Population of Antibodies, showing how they match 4 antigens A,B,C,D

that are in some ways a compromise between the 'generalist' and 'specialist' antibodies described by Forrest. For example, it is unlikely that a common sequence of jobs will be observed in all schedules, and hence the completely generalist antibody is unlikely to exist. However, it is important to find patterns that are common to as many schedules as possible. We term this the *degree of overlap* exhibited by the antibody population. Secondly, it is also advantageous to maintain sub-populations of dissimilar antibodies that match the same antigen, as a schedule may contain several sequences, each of which is common to different subset of the remaining schedules. This is referred to as the *degree of redundancy* exhibited by the antibody population. At the other extreme, highly specialist antibodies that represent patterns occuring in only one schedule are necessary to deal with the particular conditions represented by the schedule, but are less useful as a general building blocks for constructing a schedule given any scenario. The ideal population will therefore contain some niches of such antibodies. The idea is illustrated in figure 1 which shows a population of antibodies, (labelled x), and their ability to match 4 antigens, labelled A,B,C and D.

Although $PRAIS$ is loosely based on the work of Forrest and Smith, it contains several additions and modifications, each of which is now described:

2.1 Antigen Representation

An *antigen* represents the sequence of jobs occurring on a single machine m. In $PRAIS$, each antigen is an integer string of length j, where j is the number of jobs to be scheduled on the machine, and each element j_i of the string represents the identity of the ith job to be scheduled. A collection of antigens that define a particular scheduling environment is referred to as an *antigen universe*.

2.2 Antibody Representation

An *antibody* is represented by a sequence of integers, of length l, where $l < j$. l is chosen to be significantly less than j as it is expected that by using a shorter antibody it will be easier to maintain a high degree of match between antibody and antigen. Also, we expect that the common patterns will consist of short sequences of jobs.

An antibody can also contain any number of wild-card alleles, '*', to facilitate incomplete matching. A wild-card can match any job. This has the advantage that if many of the common job-sequences are shorter than the chosen antibody length l, a partially matching antibody will have high fitness. Also, it may be possible to observe patterns of the form 'a**b', i.e where the common jobs are not consecutive.

An initial antibody population is generated completely at random, with the caveat that an antibody is not allowed to contain duplicate jobs.

2.3 Fitness Function

(Smith et al., 1993), introduce the emergent fitness sharing function, a modified version of which is described below, in which the degree of generalisation or specialisation exhibited by the evolved antibodies can be controlled by altering the parameter σ, the size of the antibody population. In order to encourage the degree of *overlap* and *redundancy* exhibited by the antibodies, we have modified steps (1) and (3) of the original fitness function, by introducing the parameter τ, the antigen sample size. (The original function is obtained by setting τ to 1).

1. Choose a sample of antigens of size τ at random and without replacement.

2. Choose a sample of size σ of the antibody population, at random and without replacement.

3. Each antibody in the sample is matched against each of the chosen antigens. A match-score is assigned to the antibody equal to the sum of the result of applying match-function M to the antibody with each of the antigens.

4. The antibody in the sample with the highest match score has its match score added to its fitness. The fitness of all other antibodies remains unchanged.

5. Repeat from step (1) for typically three times the number of antigens.

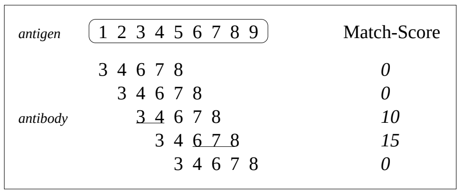

Figure 2: Possible alignments of an antibody with an antigen, and the resulting match-score of 15

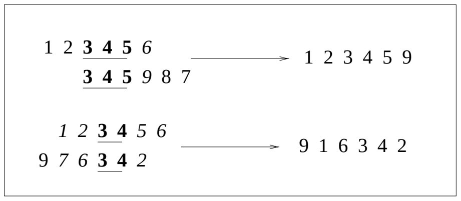

Figure 3: Overlap Crossover

2.4 The Match Function

An antibody is matched against an antigen by aligning the two strings. If the antibody is shorter than the antigen, then a match-score is calculated for every possible alignment position, and the highest score found is returned. A possible alignment is any alignment in which every gene of the antibody is aligned with every gene of the antigen. This is illustrated in figure 2.

The match-score is calculated by counting the number of matches between antigen and antibody genes in the alignment. An exact match contributes a score of 5, whereas a wild card match contributes a score of 1. This prevents the evolution of antibodies containing all wild-card genes. The reason for allowing multiple binding-sites between antigen and antibody is that a job-sequence described by the antibody may occur at any position in an antigen. This is also a feature observed in the biological immune system, where both antibody and antigen have multiple binding sites.

2.5 Parameters and Operators

A genetic algorithm based on GENESIS, (Grefenstette, 1984), is used to evolve the antibody population. Reproduction of antibodies takes place via one of three crossover operators. The operator chosen depends on the relationship between the two parent antibodies:

Order-Based Crossover (OX) — if the parents are permutations of each other, then use a permutation crossover operator — OX, (Davis, 1985).

2pt-Crossover — if the parents do *not* have any genes in common, (excluding wild cards) and the parents differ outside of a randomly chosen cross-segment, use 2pt crossover.

Overlap-Crossover — used if one parent "overlaps" the other, as shown in figure 3. In this case, align the parents so that the matching regions line up, and then read from the left most position. If only one parent has a gene at a position, use that in the child, if both parents have a gene at the position, select randomly from either parent. Continue reading from left to right until the child is of the required length. In figure 3, matching regions are underlined and shown in bold, and genes which can be chosen from either parent are shown in italics.

A mutation operator is applied to each child antibody that randomly mutates each gene with probability $1/l$.

All reported experiments were performed using a population of size 100, with the length of each antibody in the population equal to 5 jobs. Each experiment was run for 250 generations and was repeated 10 times. The mutation rate in each case was $1/l = 0.2$. The crossover rate was set to 0.7. Details of the settings for σ and τ are described in the relevant subsections.

2.6 Test Data

10 test-scenarios were generated from a base problem which contained 15 jobs. Each job had a different arrival-date, due-date and processing time, and the aim is to minimise T_{max}, the maximal tardiness of a job. The test-scenarios were produced by applying a mutation operator with probability 0.2 to each of the arrival dates given in the base problem. The mutation operator randomly changed the arrival date, with the caveat that the new arrival-date was still at least p_t days before the expected due-date, where p_t was the processing time of the job.

Satisfactory schedules were then found for each of the 10 scenarios, using a genetic algorithm described in (Fang et al., 1993), and the resulting job-sequences noted, and used as antigens. The 10 antigens define a universe denoted U(0.2).

3 Experimental Results

This section contains the results of a series of experiments performed using *PRAIS* on the test data described above. Due to space limitations, we report results here only for experiments using antibodies of size 5, i.e 1/3 the length of the antigen string. Also, we only give results for recognising patterns on one of the 5 machines, although experiments were performed for all 5 machines and will be reported in detail in forthcoming work. Experiments were performed to identify good settings for three main parameters; the antibody sample size σ, the antigen sample size τ, and the length of the antibody l.

To simplify reporting of results in the following sections, we introduce an additional concept of *binding* — an antibody and antigen are said to *bind* if the number of non wild-card positions in which the antigen and antibody agree is greater than or equal to some threshold value t_m.

3.1 Coverage of Antigen Universe by Antibody Population

Table 1 shows the average number of antigens (from the universe containing 10 antigens) that were *not* bound by any antibody, for match-thresholds t_m ranging from 2 to 5. Experiments were performed over a range of values for σ and τ, the size of the antibody and antigen samples respectively.

For certain combinations of values of (t_m, σ, τ), a large number of antigens are not bound by antibodies. This is particularly noticeable as the size of τ increases. At high values of τ, antibodies that achieve a high match-score with more than one antigen are rewarded most highly by the fitness function. However, in many antigen-universes, it may be impossible to detect common patterns between certain subsets of antigen, and hence the completely generalist antibody may not exist. Examining the antigen universe for the machine in question indicates that this is indeed the case — if subsets of 8 antigens are selected for instance, no common schemas or patterns may be found. Similarly, low values of σ also encourage generalist antibodies to evolve, so we may expect poor performance if the value of σ is too low.

3.2 Number of Antibody Niches Discovered

Recall that the purpose of *PRAIS* is to evolve a collection of diverse antibodies, each of which represents some commonly occurring pattern in the antigen-universe. The more unique patterns, i.e antibody niches, we are able to detect, then the more useful the antibodies will be as building blocks for constructing new schedules. Therefore, the final population of antibodies is examined to determine the exact number of distinct antibody niches in which the antibodies in the niche bind to antigen for a variety of values of t_m.

Table 2 shows the results obtained by summing the number of binding niches found when $t_m \geq 2$. It is clear that number of niches decreases as τ is increased, and increases as σ increases. This is unsurprising, due to the same arguments outlined in section 3.1.

When the number of niches is low, each niche must contain a large number of identical antibodies. This may ultimately be useful in the final schedule construction phase. If we consider the number of copies of an antibody in the population as analogous to a "concentration", then this concentration can be used as a measure of probability of picking the antibody when trying to reconstruct a schedule. This has strong parallels with the real immune system in which the concentration of an antibody that can bind successfully to an invading pathogen rapidly increases after the initial recognition phase, (Roitt et al., 1998).

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

Table 1: Average number of antigens (out of a possible 10) not bound by *any* antibody

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

Table 2: The Number of Antibody Niches in the Final Population when $t_m \geq 2$

3.3 Degree of overlap exhibited by the antibody population.

The two previous sections have shown that it is possible to evolve a set of unique antibodies, and also that those antibodies tend to bind to at least 1 antigen.

In order to quantify the degree of overlap exhibited by the antibody population, we record the number of

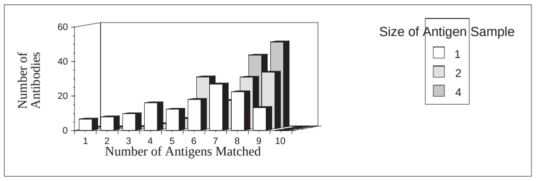

Figure 4: $t_m = 2$: Number of antibodies binding to > 1 antibody for antibody sample size $= 30$

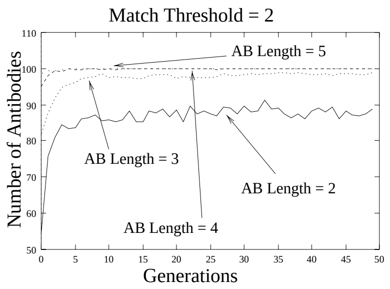

Figure 5: Number of antibodies binding to more than 1 antigen when $t_m = 2$

different antigens bound by each antibody in the population. As noted previously, the most useful set of antibodies will contain antibodies that have a high degree of general characteristics, i.e. each antibody binds to more than one antigen. Figure 4 shows the number of antibodies that bind to n antigens, where n takes values between 1 and 10. The diagram contrasts the results obtained when $t_m = 2$, using a fixed antibody sample size $\sigma = 30$ and various values of τ. Clearly, more antigens are bound at high values of τ, as expected.

It is interesting to observe how the number of antibodies binding to more than 1 antigen increases as the GA runs. Figures 5 illustrates the idea for $t_m = 2$, in which the antibody length l is varied from 2 to 5. When the match-threshold is equal to the antibody length, there is a rapid increase in the number of matching antibodies in the $t_m = 2$ case after which the number remains relatively constant. For values of $t_m < l$, there is a immediate increase at the the start of each run to a level which is maintained throughout the remainder of the experiment. The initial rise is more pronounced when t_m is significantly less than l.

4 Reconstructing Schedules from the Antibody Population

The natural immune system employs a set of combinatoric mechanisms in order to construct antibodies (or pattern detectors) from a germ-line or library of DNA chunks. Assuming that the germ-line contains n DNA chunks, and that a *minimum* of s chunks are required to construct an antibody, then there are *at least* $C = \binom{n}{s}$ possible combinations. In the case of the scheduling system, each of the n chunks of DNA can be considered analogous to a partial sequence of jobs, or more precisely, to one of the antibodies output from $PRAIS$. For a scenario in which j jobs need to be scheduled on a single machine, then examining all possible sequences of these jobs results in $j!$ possible schedules. We postulate that C_v, the number of *valid schedules* in C is $\ll j!$ for two reasons:

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

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

For instance, for the example described in section 5.1, in which $j = 15$, $l = 5$, $n = 58$, and $s = 3$, then $j! = 1.3\text{x}10^{12}$ and the lower bound on C is $\binom{58}{3} =$ 30,856. C_v of course is likely to be much less than C.

4.1 CLARISA

A immune-system model to recombine artificial DNA into antibodies in order to perform simple pattern recognition tasks has previously been implemented by the authors and is described in (Hart, 1998). This system is dubbed CLARISA, [1]. Some modifications to this basic system are made to make it suitable for recombining the antibodies from $PRAIS$ into completed schedules — the mechanisms used to combine the DNA chunks to produce schedules are described below. Three recombination mechanisms are employed, each closely related to a feature observed in the natural immune system. Input to $CLARISA$ is thus a set of antibodies, each of length l, and a partial schedule of length $l_p < j$ which must be completed. If $l_p = 0$ then the schedule is constructed from the beginning.

Simple Recombination In this method, an antibody is selected at random from the subset S_1 of the $PRAIS$ population which contains those antibodies in which every job in the antibody has not yet been scheduled in the partially completed schedule. The

[1]**CL**assification via **AR**tificial **I**mmune **S**ystem **A**nalogy

new antibody is concatenated to the end of the partial schedule.

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

Single Job Addition In order that a complete schedule can be built when the antibody population does not contain at least one instance of each of the j jobs, a single job can be selected from the subset S_3 of all jobs that do not occur in any of the antibodies discovered by $PRAIS$, and added to the end of the partial schedule.

$CLARISA$ functions given a partially completed schedule (which may be empty) and a set of antibodies. Antibodies are added to the partial schedule until it is either complete, or cannot be extended further by iteratively selecting an antibody for recombination with probability p_r for simple recombination, p_{sr} for somatic recombination and p_a for single job addition.

5 Antibody Generation by Exhaustive Search Methods

Given any antigen of length j, *all* antibodies of some predefined length l containing at most w wildcards can be generated using an exhaustive procedure, without having to resort to using a GA. For the problem described in which each universe consists of 10 antigens each containing 15 jobs, this is a tractable calculation. Actually performing the calculation to find all antibodies of length 5, that contain at most 3 wildcards, results in 423 unique antibodies being found. If this is compared to the output from $PRAIS$, we see from table 2 that the maximum number of unique antibodies found in the experiments performed is 58. Although the evolved antibodies appear to have good properties in terms of antigen coverage, overlap and redundancy, we must address the question of whether the much reduced evolved set contains sufficient information to reconstruct good schedules.

5.1 Comparison of Results

The set of 58 antibodies evolved using $PRAIS$ are used as input to $CLARISA$, which generates 500 schedules using these antibodies. The generated schedules are compared to each of the original 10 schedules in the antigen universe that was used to evolve the 58 antibodies. Table 3 shows the percentage of the 10 original schedules that were exactly reconstructed by $CLARISA$, and gives results for varying values of l_p, the length of the partial schedule that must be completed. The table compares the results generated using the output from $PRAIS$ to the results found when the 423 exhaustively generated antibodies are used instead. Each result is averaged over 10 runs of $CLARISA$, using parameters $p_r = 0.5$, $p_{sr} = 0.4$, $p_a = 0.1$.

Accuracy of Schedule Reconstruction (%)

	Length of Partial Schedule		
	7	8	9
$PRAIS$	30	70	80
Exhaustive	60	60	60

Table 3: Percentage of the 10 schedules in the antigen universe that are exactly reconstructed using $CLARISA$

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

5.2 Performance in Unseen Universes

The previous section described experiments that showed that $CLARISA$ could be used to reconstruct the antigens in the original universe, U(0.2). However, in order to be most useful, the antibodies evolved by $PRAIS$ (or produced via exhaustive search) should be useful for constructing new schedules for unforeseen circumstances. Two more universes were generated using the method described in section 2.6 by applying mutation operators with probability 0.1 and 0.3 to the original job arrival dates, to produce universe U(0.1) and U(0.3) respectively. Each new universe contained 5 antigens, and satisfactory schedules were again found using a GA as before. The antigens in each universe were then compared to 3 sets of antibodies; those by $PRAIS$ for universe U(0.2), those found by exhaustive search of U(0.2), and finally to those found by exhaustive search that contained at most 3 wild cards. The number of exact matches between the antibodies and

Table 4: The number of antibodies that can bind to at least one antigen in each universe

	Antibody Generation Method		
	PRAIS	Exhaustive Search	Reduced E-Search [a]
Universe-0.1	10	65	40
Universe-0.3	19	70	43

[a] Antibodies generated by exhaustive search with at most 3 wild cards

the antigens in each universe was computed in each case, and the results are shown in table 4.

Table 4 shows that in each case, the antibodies derived from universe U(0.2) are able to match antigens in completely different universes. Thus, the implication is that *CLARISA* will provide a suitable tool for recombining these antibodies into good schedules, given some suitable choice of recombination parameters.

6 Conclusion

We have shown that an immune system metaphor can be applied to a scheduling environment in order to rapidly produce schedules whenever environmental conditions dictate that a change from the planned procedure must occur. A GA combined with an Immune System, (*PRAIS*), was used to produce a small but efficient set of building blocks or antibodies from a set of historical schedule data. By using two parameters in the GA fitness function, σ and τ, we demonstrated that the evolution could be controlled, in order to vary both the diversity of the antibodies produced, and the extent to coverage of the antigen universe. We then showed that the antibodies evolved using *PRAIS* could be used to reconstruct the set of original schedules in universe U(0.2) — i.e. the set of schedules that the immune system had been exposed to. Furthermore, we also showed that the antibodies contained sufficient information to potentially be able to construct other schedules in universes that the immune system had not previously been exposed to. The antibodies evolved by *PRAIS* were compared to two much larger sets of antibodies produced via an exhaustive search of the antibody space. Although the *PRAIS* set was much smaller, it performed extremely well in comparison.

A second immune metaphor was used to implement a system *CLARISA* that controlled the manner in which the building blocks were used to produce schedules. The results described do not represent an exhaustive search of the parameters to *CLARISA*, which has not yet been extensively calibrated, and therefore may be considerably improved by judicious choice of values for parameter settings and the length of the input antibodies, and also by adding further recombination operators that allow mutation to occur. However, the results show that even with large antibody sets, the recombination appears efficient. There is an obvious trade-off between the number of antibodies required to adequately represent possible antigen universes, and the effort required to recombine them into accurate schedules. Further work will investigate the exact nature of this trade-off, using a variety of test problems of different sizes, and also attempt to calibrate the system more thoroughly.

Acknowledgements

The first author gratefully acknowledges the support of EPSRC, grant number GR/L22232.

References

Bäck, T., Fogel, D., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation.* Oxford University Press.

Dasgupta, D. (1998). *Artificial Immune Systems and Their Applications.* Springer.

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

Fang, H.-L., Ross, P., and Corne, D. (1993). A promising ga approach to job-shop scheduling, rescheduling and open-shop scheduling. In *Proceedings of Fifth International Conference on Genetic Algorithms,*, pages 375–382, San Mateo, CA. Morgan-Kauffman.

Forrest, S., Javornik, B., Smith, R., and Perelson, A. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211.

Grefenstette, J. (1984). Genesis: A system for using genetic search procedures. In *Proceedings of a Conference on Intelligent Systems and Machines*, pages 161–165.

Hart, E. (1998). Using an immune system model for pattern recognition. Technical report, University of Edinburgh. in preparation.

Roitt, I., Brostoff, B., and Male, D. (1998). *Immunology.* Mosby International Ltd, fifth edition.

Smith, D., Forrest, S., and Perelson, A. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149.