Producing Robust Schedules Via an Artificial Immune System

Emma Hart, Peter Ross, Jeremy Nelson
Department of Artificial Intelligence, Edinburgh University, Edinburgh, U.K
Email: {emmac,peter,jeremy}@dai.ed.ac.uk

Abstract

This paper describes an artificial immune system (*AIS*) approach to producing robust schedules for a dynamic job-shop scheduling problem in which jobs arrive continually, and the environment is subject to change due to practical reasons. We investigate whether an *AIS* can be evolved using a genetic algorithm, (GA), and then used to produce sets of schedules which together cover a range of contingencies, both foreseeable and unforeseeable. We compare the quality of the schedules to those produced using a genetic algorithm specifically designed for tackling job-shop scheduling problems, and find that the schedules produced from the evolved *AIS* compare favourably to those produced by the GA. Furthermore, we find that the *AIS* schedules are *robust* in that there are large similarities between each schedule in the set, indicating that a switch from one schedule to another could be performed with minimal disruption if rescheduling is required.

1. Introduction

The biological immune system is responsible for defending the body against pathogens and other toxins that may be harmful. It does this by producing *antibodies*, which recognise foreign molecules or *antigens*, and physically bind to them, eventually leading to their elimination. There are an almost limitless number of possible antigens, yet despite having fairly limited genetic resources, the human immune system has evolved in a manner that allows it to successfully deal with an enormous range of antigens, reacting quickly both to those antigens it has encountered before as well as to entirely new ones.

Applying the analogy to a shop-floor environment, it would extremely useful to maintain a scheduling system which was able to produce schedules which could cope with the wide range of potential situations that could occur, both predictable and unpredictable. Although in a purely deterministic job-shop, all job-arrival dates and machine processing times are known, it is easy to envisage many practical situations occurring which would require a change in the original schedules — for example, a machine breaking down, due dates of jobs change due to changing customer priorities, or jobs arriving later than planned.

Much previous work in the job-shop scheduling domain, for example [7],[1] has concentrated on producing optimal schedules that minimise some criterion, for instance turn-around time or job tardiness. However, an optimal schedule may often be extremely fragile — a slight alteration to one or more of the job or machine attributes may drastically affect the schedule. Thus, given the fluctuating nature of a real shop floor, we aim to produce sets of schedules that are resilient to changes that may occur, using an *AIS*.

To make the analogy between an immune system and the job-shop scheduling problem explicit, we consider an *antigen* to represent a set of changes that can occur that may force a schedule to change. An antibody represents the schedule itself, and the function of the immune system is to produce a set of antibodies that together cover all the contingencies defined by a set of antigens. We concentrate on a job-shop problem in which we presume that a single plan defines (j*m) operations of j jobs on m machines. Each operation has a fixed processing time, and each job has a due-date by which time it must complete which remains constant throughout all experiments. A machine can only process one job at a time, and preemption of any operation on any machine is not allowed. Each job has an associated arrival-date. We consider situations in which the contingencies the schedules must cover represent changes in the expected arrival dates of one or more of the jobs. The quality of a schedule is judged by maximum tardiness, T_{max}, given the expected job-arrival dates, i.e. the cost of a schedule is directly related to its latest job that completes after its due-date.

2. Description

In the natural immune system, the genetic material required to produce an antibody molecule is stored in 5 separate component libraries. An antibody is produced by combining a randomly selected component from each library, as show in figure 1. Thus an immune system containing l libraries, each with c components, can be used to format c^l different antibodies. The complete set of antibodies that can be

0-7803-4869-9/98 $10.00©1998 IEEE

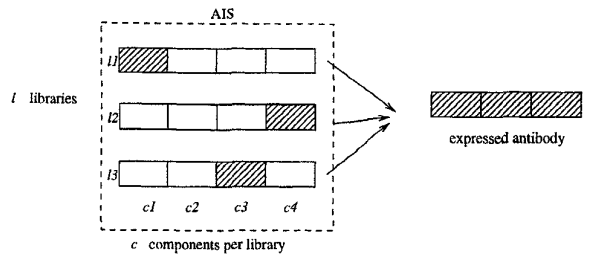

Figure 1: Expressing an antibody from an artificial immune system

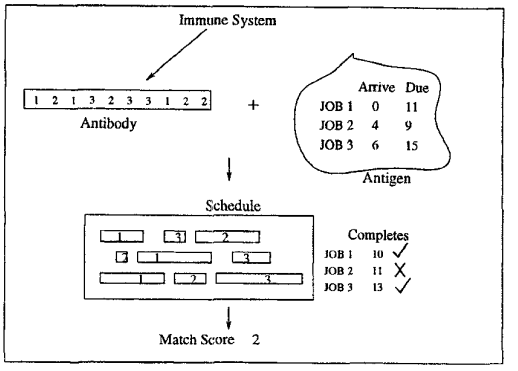

Figure 2: Operation of an Artificial Immune System for Scheduling

formed is known as the *potential antibody repertoire*. If the components in each library are genetically dissimilar, then the scope for producing a set of antibodies that together match a wide range of antigens is increased.

In the artificial immune system, *AIS*, described, an *antibody* indirectly represents a schedule. It consists of a string "abcd..." of length jm, which is interpreted as "place the 1st untackled task of the ath uncompleted job into the schedule, then place the first untackled task of the bth uncompleted job into the schedule etc.". This representation was described by [3] and was found to be very successful in tackling a wide range of scheduling problems. Each antibody is produced from an AIS which consists of l libraries, each containing c components, by randomly combining components. Each component is a string of s genes. Each of the s genes has a value in the range $(0-j)$. The values of (l, c, s) can be varied, subject to the condition that ls=jm, so that the length of the expressed antibody is equal to the number of operations.

An *antigen* describes a set of expected arrival dates for each job in the shop and hence each antigen represents one of the contingencies we wish to deal with. The complete set of antigens that an immune system may be exposed to is called the *antigen universe*. An antibody is said to *match* an antigen if the schedule it represents satisfies each of the fixed due dates, given the arrival dates defined in the antigen, and is assigned a *match-score* of 0. If any of the due-dates are exceeded, then the match-score is equivalent to T_{max}, i.e the maximum number of days a job is late. A diagram indicating the operation of an *AIS* is shown in figure 2.

3. Evolving an *AIS*

We use a genetic algorithm to attempt to evolve an *AIS*. Each individual in the population represents a single *AIS*. The total number of genes in each individual is thus lcs. Each *AIS* in the initial population is generated by assigning a random value to each of the lcs genes. For a defined antigen universe, the quality, or fitness, of each *AIS* in the population is calculated by the following procedure:

- Express N antibodies at random from the *AIS*, and decode them into schedules.
- Select K antigens at random, with replacement, from the antigen universe.
- For each of the K antigens selected:
  - Using the arrival-dates defined by antigen k, calculate the match-score of each expressed antibody.
  - Assign antigen k an *antigen-score*, equal to the best (i.e. lowest) match-score.
- Average the K antigen-scores to give an overall fitness for the individual.

The quantitative fitness value indicates how well a particular immune system is able to cope with the antigen universe in which it exists. Note that each chromosome contains a large amount of redundant information, and its fitness is based on incomplete sampling of its environment. Furthermore, selection pressure operates only on the phenotype, yet it has been shown that this is sufficient to drive evolutionary changes in the genotype, for example [5].

4. Experiments

Antigen universes, (AUs), were generated based on a benchmark problem given by Morton&Pentico in [8] — the actual problem used was jb11.ss. This problem contains 15 jobs, to be processed on 5 machines, and is known to have an optimal solution where no job arrives late. Each AU generated contained 10 antigens — an antigen was generated by

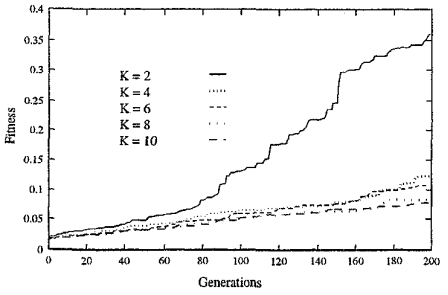

Figure 3: Generation vs Fitness for varying antigen exposure rates, at a constant antibody expression rate N=15

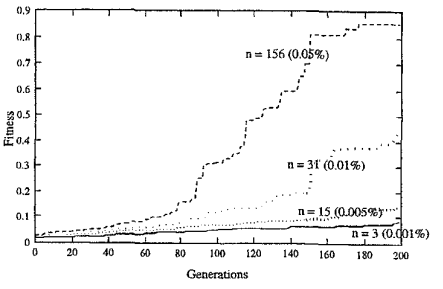

Figure 4: Generation vs Fitness for varying antibody expression rates, at a constant antigen exposure K = 2

mutating the original arrival date for each job with probability p_u to another random date, in the range (0,300), subject to the condition that the new arrival date was at least *pt* days before the due-date of the job, where *pt* was the minimum processing time required to complete the job. (Note that this method does not guarantee that the resulting conditions can lead to an optimum schedule where no job is tardy.)

A population of 100 *AIS*s was generated, with each *AIS* characterised by $(l = 5, c = 5, s = 15)$. A generational reproduction strategy was employed, with recombination performed by tournament selection with size 5, and uniform crossover. Each experiment was run for 200 generations, and repeated 10 times. The antigen universe contained 10 antigens and was generated by setting $p_u = 0.2$.

Two series of experiments were performed:

1. For a fixed antibody expression rate $N = 15$ (0.005% of c^l), vary the antigen exposure rate K.
2. For fixed antigen exposure $K = 2$, vary the antibody expression rate N.

Figures 3 and 4 show the results of these experiments. Performance clearly increases as N is increased, as would be expected with greater sampling of the genetic material available in the *AIS* genotype. Even with only 0.001% of the potential antibody repertoire expressed, some evolution of the genetic material takes place, though somewhat slowly. When 0.05% of the repertoire is expressed, evolution is rapid, and approaches the maximum possible fitness of 1.0.

On the other hand, average fitness decreases as the antigen exposure rate K is increased. This indicates that there is perhaps insufficient diversity within the immune system to cope with the number and diversity of antigens in its universe. This result conflicts directly with the work by Hightower *et al.*, [5], in which they find that *binary* immune systems evolve faster and end up with higher fitness values as the antigen expression rate is increased. Nevertheless, evolution does take place at all values of K over 200 generations.

4.1. Somatic Hyper-mutation and the Baldwin Effect

In a real-life immune system, an effect known as *somatic hyper-mutation* is observed in which stimulated antibodies produce daughter cells, in which one or more genes become mutated. The daughter cells thus have varying abilities to recognise a single antigen. Certain key mutations can lead to a significantly increased recognition ability, and hence result in an increase in the match-score of the stimulating antibody. However, these key mutations are not written back to the parent genome and hence cannot directly be passed onto future offspring. This phenomenon of passing useful characteristics down to a future generation without genetic propagation is known as the Baldwin effect, and was proposed by Baldwin 100 years ago, [6]. Hightower, [4] and Perelson, [9], investigate a modified statement of the Baldwin effect, that "learning accelerates evolution", by including an element of somatic learning during the evolution of binary immune libraries. We investigate whether the effect can speed the evolution of non-binary immune systems using a similar procedure — after a match-score for an antibody has been calculated, the antibody is mutated at random in M positions, and the match-score recalculated. If the match-score improves, then the original antibody is assigned this new match-score. The mutations are not written back to the gene library. Figure 5 shows the effect of including somatic mutation during the evolution for values of M ranging between 0 and 40. Each experiment was repeated 10 times for each value of M, using an antibody expression rate $N = 15$ and antigen exposure rate $K = 4$. The graph shows that a clear improvement in tardiness is gained by including an element of somatic learning during the evolutionary process, with the greatest gain at low values of M.

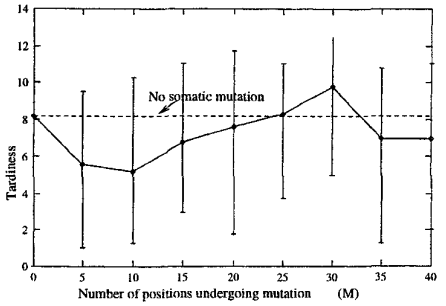

Figure 5: Effect of increasing somatic mutation rate M on tardiness

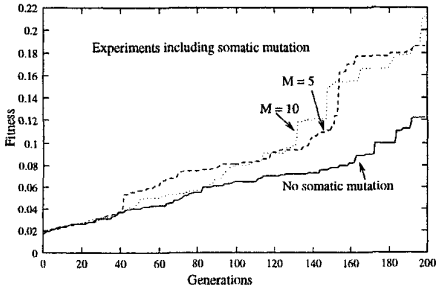

Figure 6: Generation vs Fitness for experiments using somatic mutation rates M of 0, 5 and 10

Figure 6, which compares the evolving fitness of the immune system with increasing generations for experiments that do and do not include somatic mutation, shows that including somatic mutation results in a more rapid evolution, and results in a higher overall fitness.

5. Inducing an Immune Response

Once a satisfactory *AIS* has been evolved, it should be able to recognise antigens in the universe in which it has evolved, and produce a response, in this case, a schedule. As in the real-world however, new antigens can also appear. We investigate whether the *AIS* produced via evolution can respond to antigens that exist in the universe in which it evolved, and also how successfully it can respond to the appearance of new antigens.

In each experiment described below, we simulate an immune-response to a single antigen A from an *AIS* in the following manner, which is loosely based on the response in the human immune system:

1. Express N antibodies at random.
2. Calculate the match-score for each of these antibodies, and select the antibody with the lowest match-score, AB^*.
3. Produce C clones of AB^*, by mutating each gene with probability p_m.
4. Calculate the match-score for each clone, and return the fittest, C^*.

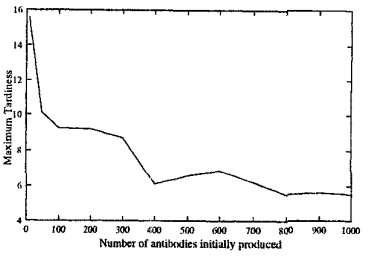

Figure 7: Effect of changing the number of initial antibodies N, on immune response. C is fixed at 1000, and p_m at 0.2

In order to quantify the success of the *AIS*, we use the genetic algorithm code produced by Fang[1] and discussed in [2] to evolve a schedule for each antigen individually, and compare the fitness of the resulting schedule, and the actual schedule itself, with those produced from the *AIS*. Fang's genetic algorithm is run for the same number of generations and using the same parameters and operators as used in evolving the *AIS*s to enable a direct comparison.

5.1. Selecting the Clone rate, Antibody Expression rate and Mutation rate

An initial series of experiments investigated the effect and importance of the choice of values for N, C and p_m in producing an response. A response was generated 100 times, and the results averaged, using a single antigen produced from the antigen universe $p_u = 0.2$. These results are shown in figures 7, 8 and 9. The fittest schedules are obtained at low mutation rates, and at high values of both N and C.

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

5.2. Recognition of Programmed Antigens

For each antigen-exposure rate, K, tested in section , the best *AIS* produced in the 10 experiments was used to try

[1] available by anonymous ftp from ftp://dai.ed.ac.uk/pub/

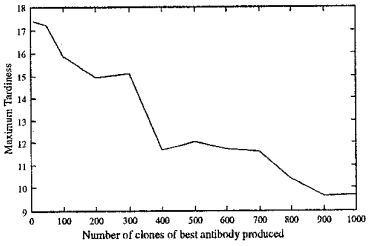

Figure 8: Effect of changing the clone rate C, on immune response. N is fixed at 100, and p_m at 0.2

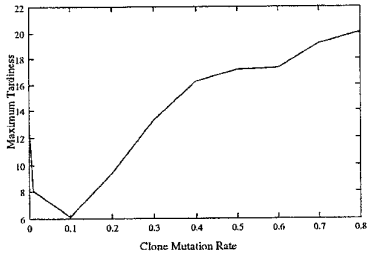

Figure 9: Effect of changing the mutation rate p_m, on immune response. N is fixed at 100, and C at 1000

and produce schedules for each of the 10 antigens in the universe ($p = 0.2$), using the procedure outlined above. Each experiment was repeated 100 times, with $p_m = 0.2$. Table 1 shows the percentage of the 10 antigens for which the tardiness of the *best* schedule found by the *AIS* was superior to that found by Fang. The figures in brackets give the corresponding percentage values for the *average* tardiness. We find that the *AIS*s evolved using a high rate of antigen exposure are generally most successful at producing good schedules, despite showing poorer overall fitness in the experiments in section 4. The trend is not completely clear however. It is also noteworthy that results produced using Fang's code required 20000 evaluations of schedules (100 individuals over 200 generations) — in the *AIS* experiments, at most 2000 evaluations are required (1000 initial antibodies + 1000 clones). Furthermore, figure 7 suggests that the number of antibodies initially produced could also be significantly reduced, perhaps to 400, as increasing the number beyond this point does not produce a corresponding increase in performance.

5.3. Response to New Antigens

Three more antigen universes were generated, with $p_u = 0.1$, $p_u = 0.3$ and $p_u = 0.5$. As above, experiments were performed to produce schedules from the 5 *AIS*s evolved in section at antigen exposure rates 2, 4, 6, 8, 10. The results are given in table 1. In general, *AIS*s evolved at a low antigen exposure (K=2) perform badly. The *AIS*s evolved at K=4 and K=8 performed well. Again there is no clear trend in the results as K is increased.

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

Table 1: Percentage of test-cases where *best* and *(average)* tardiness of *AIS* schedule was equal to or less than result found by Fang

The ability of the *AIS* to match the results produced by Fang decreases as p_u increases. Even at $p_u = 0.5$ however, using the *AIS* from $K = 8$ we are able to match the Fang results in 40% of cases, which is surprising, considering the wide diversity of arrival-dates amongst the antigens.

6. Robustness of Schedules

Ideally, we would like one schedule to cover more than one contingency, at the same time as maximising throughput through the workshop and minimising idle time. Table 1 shows that we *can* generate a schedule for each new set of circumstances — however, this may result in schedules that are significantly different from the original ones and cause much disruption if they were to be put into practice.

We introduce a measure of similarity of two schedules, notionally based on Hamming Distance. Given a schedule which describes the operation of j jobs on m machines, then we can write the schedule as a single string of length jm, where the first j alleles indicate the order in which jobs are to be processed on machine 1, the next j alleles indicate the order that jobs are processed on machine 2 etc. In the sense that two schedules are similar if jobs are processed on a machine in the *same* order in each schedule, regardless of the time that the jobs start or finish, we can then directly compare the similarity of two schedules by counting the number of places in each string that the two schedules differ. In this case, we have 15 jobs to be scheduled on 5 machines — the probability that two loci contain the same allele is 1/15 and hence on average we would expect 0.067% of the positions to be the same if the schedules were generated completely at random.

A pairwise comparison of each of the 10 schedules generated for each value p_u was performed, first using 10 schedules generated from an *AIS*, and then using 10 schedules produced using the Fang code. The average of the calcu-

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

Table 2: Hamming Separations of Schedules in Different Antigen Universes

lated Hamming Separations in each case is given in table 2.

Apart from the anomalous case of $p_u = 0.2$, we see an improvement gained by producing schedules from the *AIS*, in that the average Hamming separation is lower and therefore the schedules more similar. This is more apparent on actually examining the schedules themselves. The difference in Hamming Separation between the *AIS* schedules and the Fang schedules decreases as p_u increases, and is also very high for high values of p_u. This is probably due to the increased diversity in antigens that is obtained by mutating each arrival-date with high probability, and the resulting low probability of producing a single schedule that can cover all cases effectively.

7. Conclusion

An immune-system analogy has been applied to the real-world problem of job shop scheduling and has produced promising results. We have shown that using a genetic algorithm, we can evolve an artificial immune system starting from an initially random state, in an antigen universe that consists of a set of circumstances that could occur in a real job-shop. From an evolved *AIS*, we have successfully retrieved schedules corresponding to antigens existing in the universe that the *AIS* evolved in, and also shown that we can produce schedules corresponding to new antigens, previously unseen by the immune system. As the diversity of antigens within a universe increases, performance of the immune-system decreases — this can perhaps be counteracted by introducing more diversity into the *AIS*, perhaps by increasing the number of gene libraries or components within those libraries. This is the subject of further investigation.

In the human immune-system, a single antibody can successfully bind to more than one antigen, i.e. only a partial matching of antibody to antigen is required for the two to bind. The number of antigens that a single antibody can bind to is a measure of the *coverage* of the antibody, and is part of the reason why the immune system is so successful, despite containing a relatively small number of genes. Similarly, one schedule may cover more than one scenario, and hence current work is investigating how effectively a *single* schedule represented by an antibody is at covering *several* or *all* of the sets of conditions defined by the antigen. Whilst it is unlikely that every set of conditions can be covered by the same schedule at no cost, a satisfactory schedule may be found within a given error tolerance. Future work should also apply these techniques to a much wider range of problems, both benchmark and real-world if possible, to investigate how it scales up to larger and more difficult problems.

Acknowledgements

The first and third authors acknowledge the support of EPSRC grant no. GR/L22232.

References

[1] L. Davis. Job shop scheduling with genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 136–140. San Mateo: Morgan Kaufmann, 1985.

[2] Hsiao-Lan Fang. *Genetic Algorithms in Timetabling and Scheduling*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.

[3] Hsiao-Lan Fang, Peter Ross, and Dave Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 375–382. San Mateo: Morgan Kaufmann, 1993.

[4] Ron R. Hightower, Stephanie Forrest, and Alan S. Perelson. The baldwin effect in the immune system: Learning by somatic hypermutation. Working paper.

[5] Ron R. Hightower, Stephanie Forrest, and Alan S. Perelson. The evolution of emergent organization in immune system gene libraries. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 344–350.

[6] J.M.Baldwin. A new factor in evolution. In *Americam Naturalist*, number 30, pages 441–451. 1896.

[7] Shyh-Chang Lin, Erik D. Goodman, and William F. Punch. A genetic algorithm approach to dynamic job-shop scheduling problems. In Thomas Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 481–489. Morgan-Kaufmann, 1997.

[8] T.E. Morton and D.W. Pentico. *Heuristic Scheduling Systems*. John Wiley, 1993.

[9] Alan Perelson, Ron Hightower, and Stephanie Forrest. Evolution and somatic learning in v-region genes. availableb by ftp from ftp://ftp.cs.unm.edu/pub/forrest/v-region.ps, 1996.