Enhancing the Performance of a GA through Visualisation

Emma Hart and Peter Ross
Artificial Intelligence Applications Institute
Division of Informatics
University of Edinburgh
EH1 1HN
+44 131 650 9341
emmah,peter@dai.ed.ac.uk

Abstract

This article describes a new tool for visualising genetic algorithms, (GAs) which is designed in order to allow the implicit mechanisms of the GA — i.e. crossover and mutation — to be thoroughly analysed. This allows the user to determine whether these mechanisms are essential to a GAs performance, and if so, to provide a principled means of setting the parameters associated with them, based on a sound understanding of their effects. The use of the tool is illustrated by applying to the analysis of a job-shop scheduling problem, in order to choose effective operators, and to determine appropriate settings for them. We show that by analysing two crossover operators and a mutation operator, we can refine the choice and settings of these parameters in order to improve the performance of the GA on the particular problem chosen. When the new operators are applied to a wider range of problems of the same type, a similar improvement in performance is observed.

1 INTRODUCTION

Evolutionary computation is now widely used in a broad range of problem solving domains. It can be extremely simple to quickly set up a very naïve genetic algorithm (GA) to tackle a problem, but analysing the results to discover whether the process is efficient or could be improved is extremely difficult due to the highly multi-dimensional nature of the data, and the quantity of it that can be generated. For this reason, operators and their associated settings are often chosen through an empirical process of trial and error, or according to published rules of thumb, despite attempts to automate the procedure (for example (Corne et al., 1994),(Tuson and Ross, 1998)). Clearly it would be preferable to approach the problem in a more principled manner, using some method which allows the GA user to analyse and understand the performance of GA on a given problem, and then extrapolate the findings in a scientific way to improve the results.

We propose that *visualising* the GA process is essential to this task. Visualisation of evolutionary algorithms, with a view to understanding the search-space covered by a GA, is now an established field. For an up-to-date flavour of the area, the reader is directed towards the recent workshop, (GECCO-99,). Reviewing more of the literature reveals that the majority of proposed techniques fall into three main categories:

1. Identifying population features
2. Visualising the space searched by the GA
3. Visualising the problem search space

Population features have been visualised using population data matrices, for instance displayed in raster format, (Collins, 1999), and by allele-loci frequency histograms, (Routen and Collins, 1993), which display information about the alleles present in a generation. Schema visualisation is described in (Collins, 1998). Visualising the *search space* has perhaps received most attention. This problem is amenable to treatment by many existing standard techniques from other fields, for example principal components analysis, (Collins, 1999), and glyphs (Spears, 1999). As many of these techniques can only be applied to single generations in turn, the use of Sammon Mapping, (Dybowski et al., 1996), has been suggested, which maintains consistent spatial relationships between generations. Collins, (Collins, 1997), has recently proposed a further method, *search space matrices*, which has the advantage of only having linear complexity. Finally, the task of visualising the *whole problem search space* has been addressed by both Cartwright and Mott, (Cartwright and Mott, 1991), with fitness landscapes, and by Mattfeld, with Configuration Space Analysis, (Mattfeld, 1996).

The techniques described above all facilitate the GA user in analysing the coverage of the explored search space, in studying the convergence behaviour of the algorithm, and in better understanding the dynamics of the evolutionary processes. However, although they tackle the question as to *which* path a GA takes through a search space, and address the question of how much of the search space is covered, they do not tackle the crucial question of *how* the GA travels along that path; this is effected by differing extents by each of the operators, and may alter at different points as the evolution progresses.

Therefore, in this article we aim to present a set of complementary visualisation techniques which do address this point. It is hoped that the combination of such tools will go some way to providing a GA user with a complete analysis tool for use when designing a genetic algorithm.

We first describe the visualisation techniques incorporated in our tool, *GAVEL*[1], and then describe how they can be used to analyse and refine a GA for use in job-shop scheduling.

2 *GAVEL*

GAVEL, is an off-line tool, which transforms textual output from a GA into a format which allows easy visualisation of all the important features of the evolution, giving the GA user the ability to analyse the performance of each of the operators (via a series of 'views' and by generating graphs) as well as grasp the dynamics of the evolutionary process.

The central feature of the method we propose is to start from the 'end' of a GA run, i.e from the best solution found, and *rewind* the evolution, rather than taking the traditional approach of watching a solution being created. Thus we note the parents of the best solution, and then their parents, etc., all the way back to the initial population, and so produce the complete ancestry tree for the best solution. As well as tracing the evolution of *individuals*, we also track the evolution of every individual *gene* contained in the final solution, tracing it back through the population to the individual it originated in. The ancestry tree forms

[1] GA Visualisation of Evolutionary Links

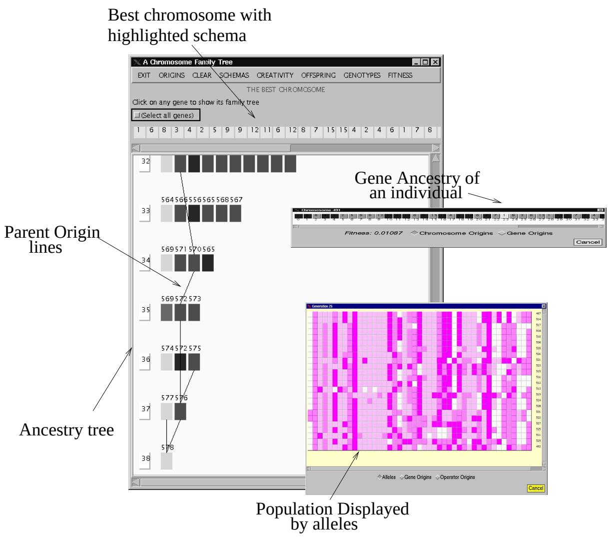


Figure 1: An Example Display from GAVEL

the central feature of our visual display tool, in which each chromosome in the tree is displayed as a clickable coloured icon. Clicking on an individual allows it to be visualised in more detail, displaying the origins and values of every gene in the individual. The shape and colouring of the tree gives an instant picture of the dynamics of the whole of the evolution of the best solution. An example is shown in figure 1.

2.1 RECORDING THE GENETIC INFORMATION

Although collecting and storing information about every gene in every chromosome encountered by a GA obviously results in enormous data-files, as we only need to utilise and hence retain information which is relevant to the formation of the best solution we can significantly reduce this problem. For example, running a GA for 50 generations with a population size of 50, and chromosome length 32 results in a *GAVEL* datafile of only 200KB, which is extracted automatically from the output of a run — this information is referred to as the *reduced population* in the remainder of this paper. Viewing only those chromosomes contributing to the ancestry tree of the best chromosome also may unmask valuable information about the dynamics, otherwise hidden in the information collected when considering the complete GA run. For example, consider figure 2 which compares the fitness vs time graphs for the complete population to that of the reduced population in the ancestry tree for two different GAs on a job-shop problem — in both cases, the GA finds the same optimum solution, but clearly the dy-

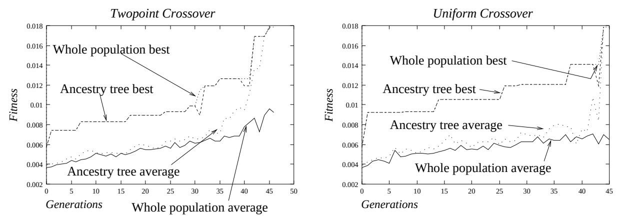


Figure 2: Comparison of fitness of reduced and complete population

namics are significantly different in each case.

2.2 USING COLOUR TO SIMPLIFY THE DISPLAY

Using the ancestry tree as the basic display, we use colour to impose different views of the data on the chromosome icons in the tree. This approach is chosen in order to surmount some of the traditional difficulties associated both with viewing multi-dimensional data, and with making sense of vast amounts of text in the form of population-data matrices. Colour coding is applied to both categorical alleles and real/integer valued alleles with the use of *colour shading*. Categorical alleles are easily represented by assigning a unique colour to each allele. For alleles in which there is a clear ordering between possible values, then a *shading* scheme is used, in which the alleles are shaded in varying degrees of a single colour. The usefulness of this approach decreases as the number of *different* alleles that need to be represented increases, as we approach the limit of how many shades are visually distinguishable to the human eye. However, in many cases it is sufficient to give a visual impression of allele distribution.

We also make extensive use of colour and highlighting to distinguish salient properties of the chromosome icons that allow the data to be viewed in several ways. These features are described detail in the following sections. From each 'view' of the data, we can also automatically derive a series of graphs, which are plotted using an external plotting program that give important information about the productivity of the genetic operators and the dynamics of the evolution.

2.2.1 Ancestry View

The complete ancestry of any individual can be traced on the tree by clicking on that individual. This causes lines to be drawn from each ancestor to its parents, back to the initial generation. Alternatively, all offspring of any individual can be highlighted in colour. Also, the lineage of any individual *gene* can also be traced on the tree, by clicking on the relevant gene. This causes lines to be drawn on the connecting all the individuals the gene has appeared in, and highlighting the generation the gene originated in.

2.2.2 Operator View

The chromosome icons can also be highlighted with a colour scheme that indicates *how* the chromosome arrived in the generation. Four colours are used: the first indicates if the chromosome was a product of crossover, the second that the chromosome was a product of crossover followed mutation, the third if it resulted from mutation only, and finally the fourth if the chromosome was copied from the previous generation. The system also allows graphs depicting *operator productivity* and *operator fitness* to be generated from this view. We define the *productivity* of an operator o in this case to mean the fraction of chromosomes in a generation of ancestors that were produced as a result of applying the operator to members of the preceding generation. *Operator fitness* is defined as the average fitness of those chromosomes in a generation of ancestors that were produced as result of applying operator o to the preceding generation.

2.2.3 Fitness View

Each individual icon can be colour shaded according to its relative fitness value (i.e in relation to the worst and best fitness in the GA run). Icons in each generation are arranged in order of decreasing fitness. Alternatively, the user can highlight all chromosomes whose *phenotypic* fitness satisfies a user-supplied constraint via a dialog window. There is also an option to highlight all chromosomes with an equivalent *genotype* — this is particularly useful for problems in which multiple genotypic representations can lead to the same phenotypic fitness value.

2.2.4 Origin View

The icons in the ancestry tree can be coloured according to their *origin*, i.e as to whether the individual was created in the generation it appears in, whether it was copied from a previous generation, or whether it was created in the current generation but does not represent a newly discovered point in the search space. Individuals which exist in a generation due to the operation of the 'elite' mechanism can also be also highlighted if this mechanism was used in the GA.

2.2.5 Schema View

Finally, individuals containing any given schema can be highlighted on the tree. Schemas are easily entered via a clickable interface.

2.3 ADDITIONAL FEATURES

Additionally, the tool allows any population of chromosomes in a single generation (or a single individual) to be viewed as a colour matrix. Again the 'view' of the matrix can be changed, to either indicate *allele values*, *gene-origins*, or *operator-origins*. If gene-origins is selected, then alleles are shaded according to the generation they first appeared in, giving a view of how mutation affects the overall performance of the GA. If operator-origins is selected, then the colouring shows either which parent the gene was passed from if it appeared as a result of crossover, or that the gene appeared as the result of a mutation. Finally, $GAVEL$ can automatically generate graphs from any of these views, which can then be plotted using an external plotting tool.

3 ENHANCING THE PERFORMANCE OF A GA ON THE JSSP

We consider an example of a job-shop scheduling problem, in which 15 jobs have to be assigned to each of 3 machines in some pre-determined order for processing, with the aim of minimising the maximum tardiness of the job completion times. The GA uses an integer representation, (described by Fang in (Fang et al., 1993)), in which a value of a at position i means "place the first untackled task of the ath uncompleted job into the earliest place where it will fit in the schedule". The list of jobs is considered as circular and hence all chromosomes represent legal schedules. We show how the performance of the GA originally described by Fang can be improved by judicious selection and setting of operators and rates. The improvements are made in a scientific manner, rather than simply based on a trial and error approach.

The following analysis illustrates two runs of the GA, both of which used the same random number to generate the initial population. All parameters in the two experiments are identical with the exception of the choice of crossover operator — experiment (a) uses two-point crossover, and experiment (b) uses uniform crossover. The random number seed of the experiments illustrated was selected so that both experiments resulted in a solution of the same fitness. (Ex-

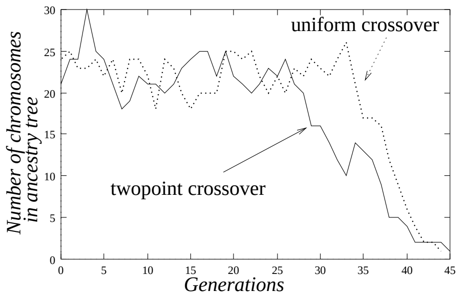


Figure 3: Number of chromosomes in ancestry tree at each generation

tensive experimentation confirmed that other seeds gave similar results). In each experiment, a population of size 50 was used, with a crossover rate of 0.8. Order mutation, (which simply swaps two randomly chosen alleles), was used in both cases, and was applied with probability 0.5 to every child chromosome in the new population, whether produced as a result of crossover or by simple copying from the previous generation. The results of the analysis of each experiment are presented in the next sections.

3.1 ANALYSIS OF THE SEARCH PATH

Firstly, the ancestry trees for each case are generated. To save space, we present a simple 2D graph derived from each tree showing the number of chromosomes contributing to the ancestry tree at each generation in each case. This is illustrated in figure 3. The total population size is 50 in both experiments (a) and (b) — in each experiment roughly 40% of the total population is involved in generating the final solution until around generation 26. The number of ancestors at each generation than rapidly tails off in the case of two-point crossover, suggesting that either the search has found a particularly fit area of the search space, or perhaps that population diversity has been reduced so far that the search remains stuck in a local optimum. In the uniform crossover case, roughly half the population still plays an active part in generating useful chromosomes for another 10 generations, before again the number of ancestors drops rapidly as the search homes in on a solution.

Thus, despite starting from the same initial population and discovering an identical local optimum at the end, the two different GAs have obviously taken very different paths through the search space. In order to understand how each path was directed by each operator, (and thus determine which operators are likely to be the more effective across a wide range of different problems of the same type), we use *GAVEL* to explore in depth the *productivity* of each operator, and to attempt to confirm some of the tentative explanations suggested in the previous paragraph.

3.2 INVESTIGATING THE PRODUCTIVITY OF THE OPERATORS

Figure 14 shows an example of the ancestry, highlighted using the *operator* view. As the detail is difficult to understand in black-and-white, we have again derived 2D-graphical versions of the data, using *GAVEL*. In the following diagrams, the percentage of the total number of ancestors at each generation which have been derived from each of the four generation mechanisms is plotted for each generation. Figures 4, 5, 6, and 7 contrast the results for the uniform crossover and two-point crossover cases, showing both the *productivity* of each operator, i.e how many chromosomes were generated by the operator, and the *relative fitness* of those chromosomes generated by the operator.

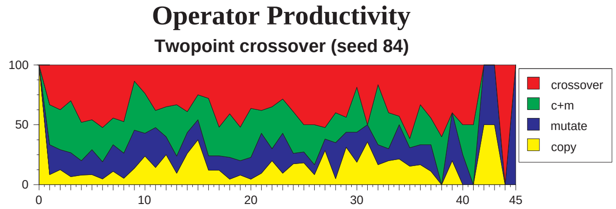


Figure 4: Operator Productivity for 2pt Crossover

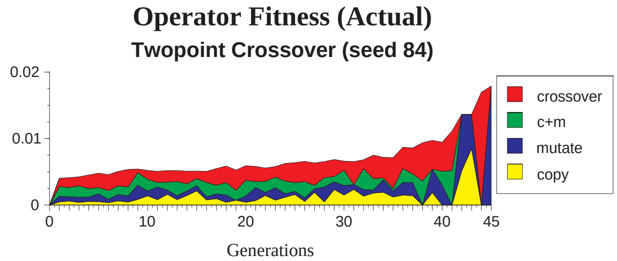


Figure 5: Operator Fitness for 2pt Crossover

Two Point Crossover In this experiment, the productivity of the mutation operator seems high, both used on its own and combined with crossover. This is especially apparent in the final few generations. This evidence points to a lack of diversity in the population, making mutation the more powerful operator. This can be checked using the population viewing facility of *GAVEL*.

Uniform Crossover In this case, mutation that is simply applied to copies of solutions from previous generations does not often result in ancestors of the best solution, though mutation of chromosomes produced by crossover appears productive. In this case, crossover appears to play the most significant part in discovering new chromosomes, especially in the final few generations.

3.3 COMPARISON TO THE EXPECTED RESULTS

In each experiment, as the crossover operator is applied with probability 0.8, we expect approximately 80% of each generation to originate as a result of a crossover operator, and similarly as mutation is applied with probability 0.5, approximately 50% of each generation to have originated as a result of a mutation operation (either applied to a copy or to a product of crossover).

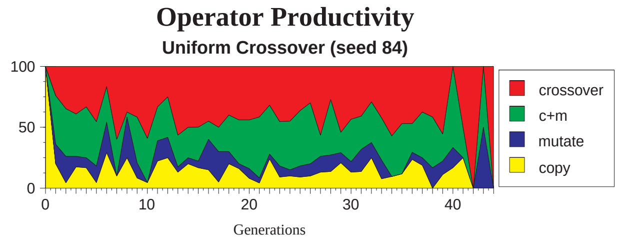


Figure 6: Operator Productivity for Uniform Crossover

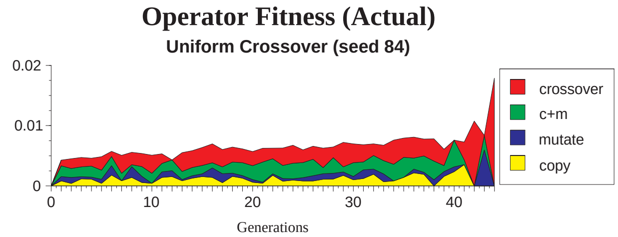


Figure 7: Operator Fitness for Uniform Crossover

We can thus plot the *deviation* in productivity from this expected value at each generation by subtracting the *observed* operator productivity from the *expected* productivity. Hence, a GA that adhered exactly to the model would be expected to show a deviation of 0 at each generation x, with a maximum positive deviation of (1-0.8) = 0.2 for crossover, a maximum positive deviation of (1-0.5)=0.5 for mutation, and corresponding maximum negative deviations of -0.8 and -0.5 for crossover and mutation respectively. The resulting graphs are shown in figures 8, 9 10, and 11. Table 1 summarises the percentage of generations in which the operator productivity is within 10% of the expected value, or is greater than/less than 10% of the expected value. In the uniform crossover experiment, it can be seen that the crossover operator is generally productive, i.e it is within 10% of expected or greater than this value in more than half the generations. In both the uniform crossover and 2pt crossover experiments, mutation does not perform well, although in the 2pt-crossover experiment it does perform significantly better than expected in 22% of generations.

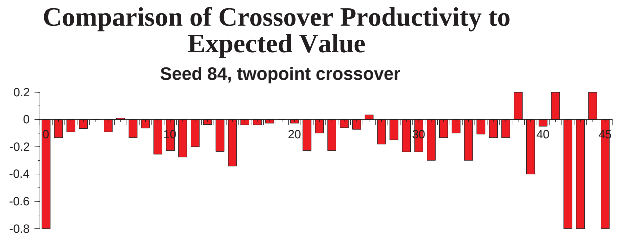


Figure 8: Comparing Crossover Productivity for 2pt Crossover

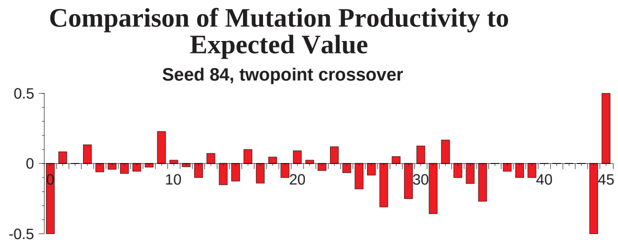


Figure 9: Comparing Mutation Productivity for 2pt Crossover

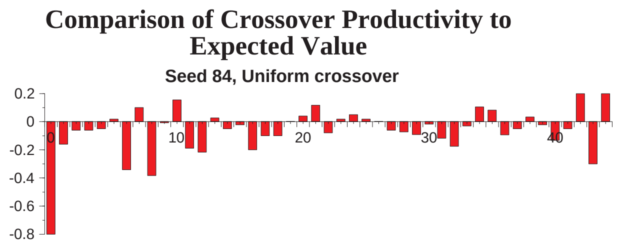


Figure 10: Comparing Crossover Productivity for Uniform Crossover

3.4 REFINING THE OPERATORS

Choice of Crossover Operator We thus conclude that crossover appears much less useful in the 2pt-crossover experiment, which seems to rely on mutation

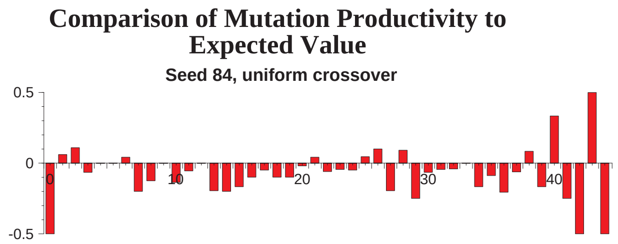


Figure 11: Comparing Mutation Productivity for Uniform Crossover

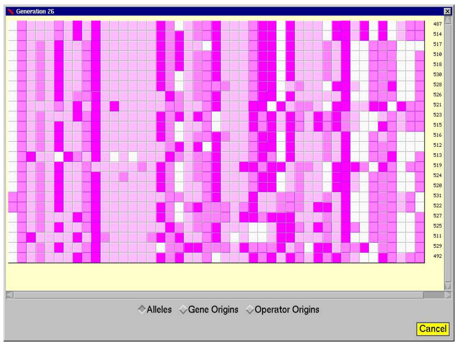
to produce the final solution. This can be partly explained by comparing population diversity in the two experiments. Consider figure 12 which compares the population diversity at generation 26, where the relative sizes of the ancestor trees begin to diverge. Each row represents a chromosome, and therefore each column a locus. We see that in the 2pt crossover experiment, there is much less diversity at the *front* of the chromosomes than when using uniform crossover, figure 12. This is critical for this particular problem representation, as the placement of jobs early on in the schedule is crucial, and much more likely to have an impact on the quality of the final schedule than jobs placed towards the end of building the schedule. The genes near the end of the chromosome have much less impact as there are so few jobs to place that the actual gene value has less meaning. As uniform crossover appears to maintain more diversity for longer, we infer that using this particular representation, uniform crossover will be a more productive operator across a wider range of problems and initial population distributions.

Selecting a Mutation Rate Table 1 showed that mutation is not often useful when used in conjunction with the uniform crossover operator. Possibly the design of the operator is at fault, or the mutation rate is incorrectly set. Figure 13 depicts the origins of each gene in the final solution. We see that 16/45 of the genes, i.e approximately 6%, originated via a mutation. Thus, although they may not be occurring very often, the mutations appear to be useful when they do occur. (In this diagram, the darker the colour, the later the mutation occurred). Secondly, consider figure 14. This shows a section of the ancestry tree, with each chromosome coloured according to its origin, and also traces via blue lines the origin of each gene in the final chromosome back through each generation to the chromosome and generation the gene originated in. The left hand box of each horizontal line which indicates the generation number is highlighted

Experiment	Operator	Within 10% of Expected Value	> 10% of Expected Value	< 10% of Expected Value
2pt Crossover	Crossover	30%	7%	63%
2pt Crossover	Mutation	30%	22%	48%
Uniform Crossover	Crossover	51%	16%	33%
Uniform Crossover	Mutation	31%	16%	53 %

Table 1: Deviation of Actual Operator Productivity from Expected Operator Productivity

Twopoint crossover



Uniform crossover

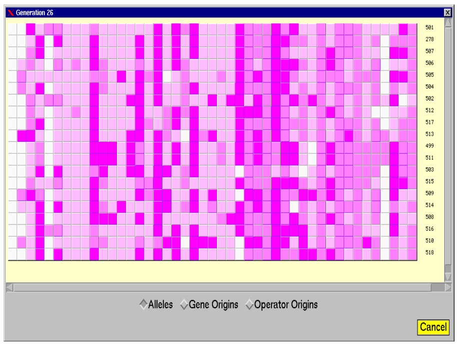


Figure 12: Comparison of Allele Diversity at Generation 26 for Different Crossover Operators

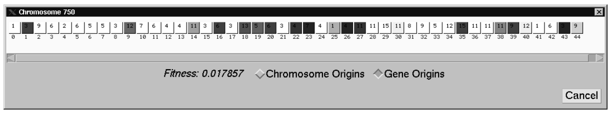


Figure 13: Origins of Genes in Final Solution

Figure 14: Ancestor Origins and Gene Origins

if a gene in the final solution originated in this generation. Notice that many of the origin lines trace paths through chromosomes that originated via mutations, but that also, these chromosomes are seldom highly fit compared to other members of their generation. This analysis suggests that mutation does have an important role to play, and that therefore better results may be obtained by increasing the rate. Hence, the mutation rate is increased to 0.7, which should tend to encourage more diversity and hence further enhance the effects of crossover. Table 2 gives the results of 3 experiments, each of which was repeated 50 times, which compare the performance of the different combinations of operator and mutation rates for one problem. The experimental results backup the conclusions drawn from the visual analysis of the problem, i.e that uniform crossover is superior to two-point crossover, and a high mutation rate is helpful. Further experimental work (not shown) confirms that the results hold over a wide range of test problems of differing lengths, and not just the single problem shown.

4 FURTHER WORK

As well as analysing operator performance, we have also used *GAVEL* to examine the phenomena of competing-conventions using this problem. This is often cited as a problem for GAs which use indirect representations. We used the *fitness view* to highlight all chromosomes in the ancestry tree of equivalent *pheno-*

Crossover Operator	Crossover Rate	Mutation Rate	Mean Value	Std. Dev
Twopoint	0.8	0.5	66.9	13.98
Uniform	0.8	0.5	62.6	13.46
Uniform	0.8	0.7	59.92	11.99

Table 2: Average Value of Objective Function (T_{max}) over 50 Experiments

type, and then also those of equivalent *genotype*. This showed that many different genotypes often existed for a given phenotypic fitness value in the ancestry tree. However, this did not seem to be problematical for the GA — in fact, it appears to make the problem easier, as the GA does not have to search for a 'needle in haystack', but simply one of many different ways of representing the same solution.

5 CONCLUSION

We have shown that visualisation and graphical analysis of the *effects* of the operators used by a GA can provide a mechanism for understanding their performance, and hence improving them in a principled manner, rather than by trial and error. Our tool is intended to complement the many other tools already available for visualising the space searched by a GA.

There are many problems inherent to visualising a GA due to the vast quantities of data generated and the multi-dimensionality of the data. Our method of visualising only those chromosomes relevant to the formation of the best solution addresses the quantity of data problem to some extent, and the format of the tree, combined with colouring highlighting, allows data to be visualised compactly at differing levels of resolution. However, the approach is still limited by the population size of the GA, and the number of generations required to find a solution, as the display becomes difficult to read if the user is required to scroll both left and right and up and down. A partial solution to this is to use the *GAVEL* display to visualise only sections of the data, whilst relying on the derived graphs to analyse the complete run.

ACKNOWLEDGEMENTS

The first author gratefully acknowledges the support of EPSRC grant no. GR/L22232.

References

Cartwright, H. and Mott, G. (1991). Looking around: Using clues from the data space to guide genetic algorithm searches. In *Proceedings of Fourth Annual Conference on Genetic Algorithms*, pages 108–114.

Collins, J. (1999). Visualisation of evolutionary algorithms using principal copmonents analysis. In Wu, A. S., editor, *GECCO Workshop Program*, pages 99–100.

Collins, T. (1997). Using software visualiation technology to help evolutionary algorithm users validate their solutions. In *Proceedsings of Seventh International Conference on Genetic Algorithms*, pages 307–314, East Lansing, MI, USA.

Collins, T. (1998). Understanding evolutionary computing: A hands on approach. In *Proceedings of IEEE World Congress on Computational Intelligence, (ICEC)*.

Corne, D., Ross, P., and Fang, H.-L. (1994). Fast practical evolutionary timetabling. Technical report, Department of Artificial Intelligence, University of Edinburgh.

Dybowski, R., Collins, T., and Weller, P. (1996). Visualisation of binary string convergence via sammon mapping. In Angeline, P. and Baeck, T., editors, *Fifth Annual Conference on Evolutionary Programming (EP96)*, pages 377–383. MIT Press.

Fang, H.-L., Ross, P., and Corne, D. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling and open-shop scheduling problems. In *Proceedings of Fifth International Conference on Genetic Algorithms.*

GECCO-99. Visualisation workshop. GECCO '99. Orlando, Florida.

Mattfeld, D. (1996). *Evolutionary Search and the Job Shop.* Physica-Verlag.

Routen, T. and Collins, T. (1993). Visualisation of A.I techniques. In *Proceedings of the International Conference on Computer Graphics and Visualisation, (COMPUGRAPH 93).* ACM Press.

Spears, W. (1999). An overview of multidimensional visualisation techniques. In Wu, A. S., editor, *GECCO '99 Workshop Program*, pages 104–105.

Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184.