Using Software Visualization technology to help Genetic Algorithm Designers

Trevor Collins †
The Knowledge Systems Group
The Knowledge Media Institute
The Open University
Walton Hall
Milton Keynes UK
MK7 6AA

Abstract

This work is part of a three year PhD project to examine how Software Visualization (SV) can be applied to support the design and construction of Genetic Algorithms (GAs). A user survey carried out at the start of this project identified a set of key system features required by GA users. A visualization system embodying these features was then designed and a prototype built. This paper describes what genetic algorithms are and how they can be applied. It then reviews some of the survey results and their impact on the design of the visualization system. The paper concludes with an exploration of how the resulting prototype may be evaluated.

† email - t.d.collins@open.ac.uk, www - http://kmi.open.ac.uk/~trevor/

1 Introduction

Genetic algorithms (GAs) are a robust type of search algorithm. Designed by John Holland at Ann Arbour, Michigan in 1970 they are based upon the evolutionary principle of "survival of the fittest" [8]. As a search algorithm operating in a symbolic domain, a GA generates a large quantity of search data, from which a near optimal solution emerges. The user's tasks in designing genetic algorithms are to:

1. Define a way of representing a problem state as a string of numbers, known as a "chromosome",

2. Construct a function capable of rating problem states (i.e. chromosomes) and returning a fitness score (typically normalised between zero and one) and,

3. Set-up the genetic algorithm by choosing a set of selection and reproduction components based on the problem representation used.

Only by monitoring the genetic algorithm during it's search can a user make an informed decision about the quality of the solutions found. Software Visualization (i.e. techniques such as cinematography, graphic design, typography and animation [9]) has been used in other domains such as prolog program development and scientific systems design, to support the design and development processes. Applying Software Visualization technology to the GA design process is put forward as a means for supporting the user tasks described above and the search monitoring necessary for quality assurance.

Section two of this paper provides a description of some GA terminology and an overview of how genetic algorithms search for solutions to user defined problems. Section three gives an overview of a survey carried out in order to gain an insight into the problems found by people designing GAs, their opinions on some sample visualization and interaction methods, and their ideas on how the work of a GA designer could be made easier. This includes a description of the survey mechanism used, a summary of the responses given and their impact on the resulting system design. Section four, the final section, presents a screen view taken from the resulting system design and examines how such a system could be evaluated.

2 Genetic Algorithms

In order to represent a problem state problem domain information is translated by the user into a symbolic representation (generally using either binary or decimal numbers) on which the genetic algorithm acts. This data takes the form of a string of numbers referred to as a "chromosome", the numbers' position in the chromosome is called the "locus" and the chromosomes' symbol values are called "alleles" (see figure 1).

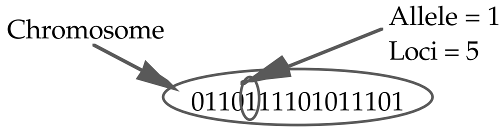

Figure 1 - An illustration of how a problem state can be represented as a string of numbers suitable for use within a GA.

An initial population (generation 0) is made up of chromosomes whose alleles have been generated using a random number generator. These chromosomes then “evolve” to form the next generation (generation 1). Evolution has two steps; "Selection" and "Mating". Selection for mating and is biased by the chromosomes’ fitness values to the extent that a chromosome with a high fitness value has a better chance of mating than one with a low fitness value.

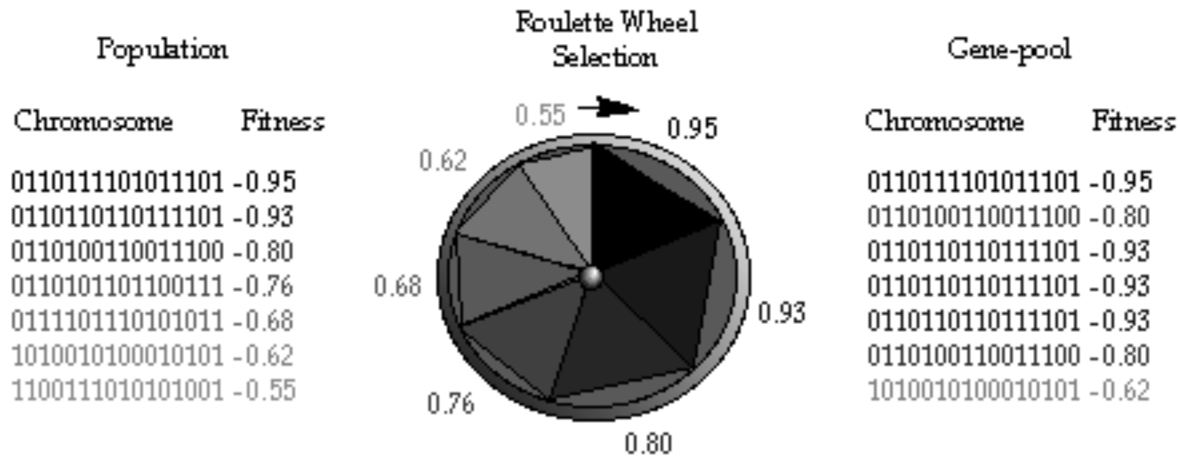

Figure 2 - An example of roulette wheel selection in which each chromosome’s chances of selection for mating is biased by the fitness rating (a grey scale is used to associate chromosomes with the appropriate roulette wheel section, the size of which is defined by the chromosome’s fitness).

One example selection method is "roulette wheel selection" (figure 2, above). Here the population chromosomes are represented as roulette wheel sections, with the size of each section being determined by the magnitude of the corresponding chromosome's fitness rating. In order to select a chromosome for mating, the roulette wheel is spun and whichever section the ball lands in determines which chromosome is selected.

Once the chromosomes for mating have been selected the second evolutionary step i.e. mating can begin. Mating is carried out by the application of "genetic operators", these are processes taken from genetics which change the features of existing chromosomes to produce new chromosomes. Two typical examples are described here; "single point crossover" and "mutation". Single point crossover is were two chromosomes are split at a random locus, the chromosome sections are then crossed and finally re-connected. As a result the two new chromosomes (the children) carry some genes from each of the two older chromosomes (the parents).

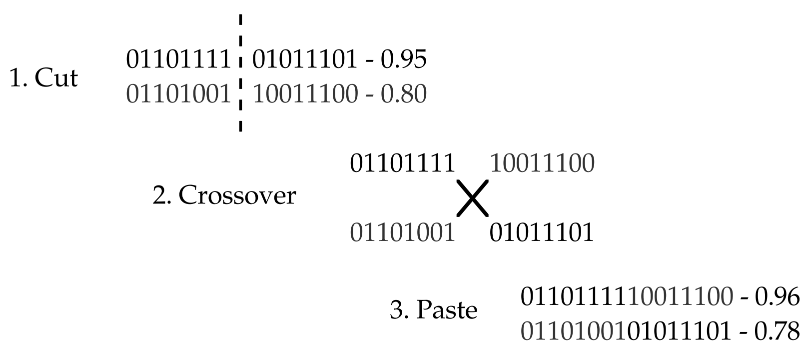

Figure 3 - An example of the three steps involved in single point crossover in order to create two new chromosomes.

As can be seen in figure 3 above the two selected chromosomes are cut at a random locus (in this case locus eight), the two sections are then crossed, and finally these are pasted together to produce two new chromosomes. As is shown in this example single point crossover can create both better and worse chromosomes.

Another example genetic operator used in chromosome mating is mutation. Mutation in GAs can be controlled by the user altering the algorithm’s mutation rate (i.e. the probability of a random allele being introduced instead of a parent’s allele).

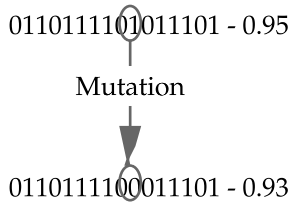

Figure 4 - An example of how mutation can be applied to a chromosome in order to alter its features.

The example given in figure 4 shows the occurrence of mutation on a randomly chosen allele (locus 10). The allele's original value has been flipped in this case from a one to a zero. These selection and mating steps are repeated until an acceptable solution is evolved. Typically the user defines an acceptable fitness rating and the algorithm repeats the evolutionary process until an individual is found with an equivalent (or better) fitness rating. Further information on GAs can be found in [8] , [5], or, [11].

3 GA User Survey

A survey of GA designers was carried out in order to gain an insight into the design process involved in applying GAs. This was then used as a basis for the design of a GA visualization system. This section describes the survey method used, gives an overview of the survey design, and explains some of the findings extracted from the survey responses.

3.1 Survey Method

An electronic questionnaire was used as a survey tool to gain an insight into the work of GA users. This was posted on three Newsnet bulletin boards known to be of interest to GA designers; comp.ai, comp.ai.alife and comp.ai.genetic. A HTML version was posted on the world wide web with an associated link from an evolutionary computing web site. Direct emails containing the questionnaire were sent to several GA researchers and research groups.

3.2 Questionnaire Design

The electronic questionnaire was split into two main sections; a background section, and a query section (an additional question seeking permission for future contact was added to the end of the questionnaire). The background section was made up of four simple questions that invited the reader to indicate their experience of genetic algorithms in terms of both time and areas of application, their reasons behind using genetic algorithms and the computer environment/s they used. These background questions were intended to provide some possible structuring methods for the responses received, such as the number of years of experience, or the problem domain being examined.

The query section made up the body of the questionnaire. This was intended to; identify the users difficulties, to suggest some possible GA visualizations, and to examine how interaction could be best used to support the user. As previously mentioned the aim of any Software Visualization system is to aid human understanding, this is only possible if the system provides support for the users difficulties, presents the information in a salient manner and provides an appropriate amount of freedom for the user to explore and examine the information in a useful way. The query section provided a means for identifying these features:

Questions five and six asked what if anything did the reader find difficult about setting up and evaluating a genetic algorithm. Questions seven through to nine enquired after the possible uses of visualization for directly and indirectly illustrating a GA. Questions ten and eleven asked about the use of interaction to aid in the exploration of the GAs search in the problem space. The reader was asked to comment on some possible interaction opportunities and to suggest any alternative opportunities they considered beneficial. As a final sub-component, question twelve asked for any other suggestions on how GAs could be made easier to use.

As well as the cover page message prefixing the questionnaire, a single paragraph presenting a brief introduction to each query section component was included within the questionnaire. This was intended not only to guide the reader in their interpretation of the questions but also to encourage a more detailed response. The questionnaire itself can be found on the World Wide Web (URL http://kmi.open.ac.uk/~trevor/Quest1.html).

3.3 Questionnaire Results

Nineteen completed questionnaires were received from experienced GA users. This section gives a brief summary of some of the questionnaire results (a more complete report is given in [4]). Some of the difficulties found by GA users and system features intended to solve them are discussed, then some of the respondents comments on system features suggested in the questionnaire are described, and finally the section closes with an explanation of one of the system features requested by a respondent.

User Difficulties

As mentioned above one of the prime purposes in carrying out this design survey was to identify the difficulties that users have in designing genetic algorithms. Two of these are described in the following sub-section.

Problem: Defining the problem representation and evaluation function

The first two questions in the query section of the questionnaire asked the respondents to describe what difficulties if any they found with defining a suitable problem representation and constructing an effective evaluation function, some typical responses were as follows:

> "The really tricky issue is designing the representation and the operators, not controlling or visualizing the GA during running."

> "This *[defining the problem representation]* is to my mind the most important step in any algorithm, perhaps more important than the choice of algorithm."

> "This *[producing the evaluation function]* can be a difficulty in many scientific problems; scaling is often necessary to ensure the algorithm does not concentrate on one variable and neglect others. Usually we find that in principle it is not too difficult to construct a suitable function, but often it must be refined once we know the behaviour of the algorithm."

Proposed Solution: A problem representation and evaluation function editor

One possible solution to this problem would be to provide the user with an editor in which they could construct a representation by defining the real-world problem features and then identifying how they map onto a string like structure. An editor of this kind could also support the construction of the evaluation function, as the contribution of each string component (i.e. "allele") would be made explicit and could be visually present throughout the evaluation function's development. Any test cases (i.e. example chromosomes) could be evaluated and both their fitness rating and problem specific description returned.

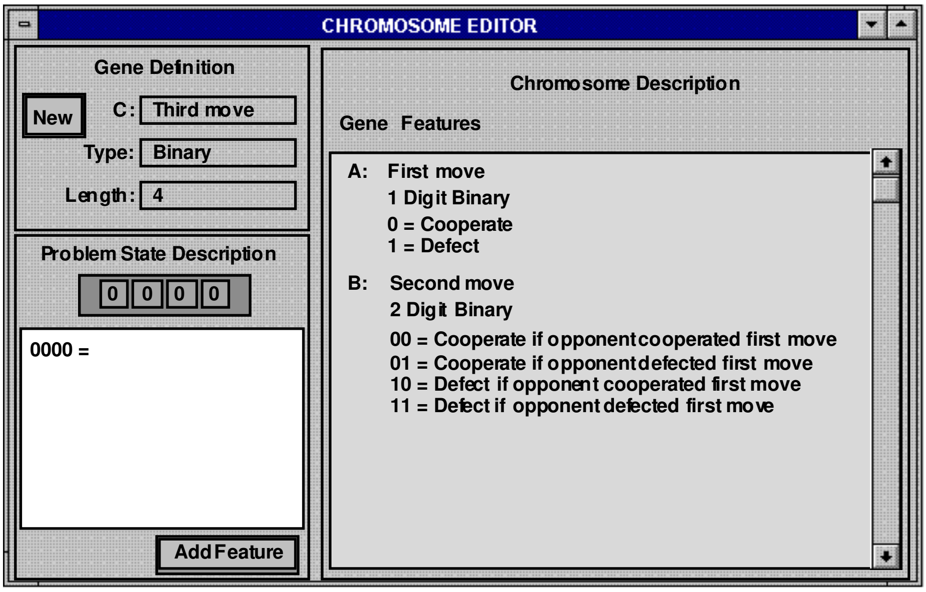

Figure 5 - An example of a problem representation editor used to input problem-specific information about a representation of the Prisoner's Dilemma game. A description of the prisoner's dilemma game is given below.

Figure 5 above illustrates how some problem-specific information could be input by the GA designer. In this way the system could provide feedback within the context of the problem being tackled. In this case an example of the Prisoner's Dilemma game. This is a two player strategy game in which each player plays the part of a prisoner who must decide either to co-operate with the other prisoner and tell the police nothing of their partnership in crime, or to defect and tell all. The four possible results correspond to different penalties against each player. The aim is to get away with the minimum penalty. In this example the user constructs the problem representation by adding new genes, each gene identifies a possible move. The first two genes (i.e. first two moves) have already been specified in the example above, and these are illustrated on the in the Chromosome Description section (top right). The user has started to define the third gene (top left) and is in the process of identifying a set of problem state descriptions (bottom left).

Problem: Choosing the algorithm's parameters

Another question raised in the user survey asked the users what, if anything, they found difficult about selecting suitable parameters for the GA, i.e. the population size, the mutation rate etc:

> "This is a real crusher, this is where your package would save a lot of time. Setting parameters is an agony for me. Every time I run the thing it takes more than a day, at the end of which all I know is that the run didn't work. It would be nice to be able to watch the run and monitor population diversity, and population movement. To some extent, the setting of parameters is _irreducibly_ hard. There are theoretical methods for setting them, which work when you know a lot about the problem i.e. it is a toy problem."

Proposed Solution: The provision of interactive parameter control

The ability to make parameter corrections without restarting the GA may well be useful. For example, as an algorithm converges toward a solution the user may wish to reintroduce some diversity into the search, in order to verify that the convergence is not premature. This could be achieved by increasing the rate of mutation during the algorithm's execution and then reducing it again once the population had become sufficiently diverse. A parameter dialogue in which the user could "tweak" their algorithm would provide this form of interactive control.

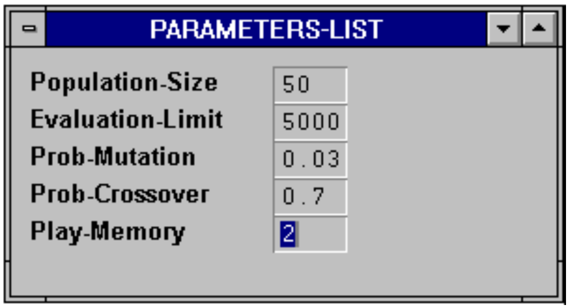

Figure 6 - An example of a parameter dialogue box.

An example of a dialogue box used for editing an algorithm's parameters is shown in figure 6 above, here the user can select and edit any of the algorithm's parameters.

Suggested Features

In addition to trying to identify the areas of difficulty in GA design, the questionnaire also asked the respondents to think of some advantages and disadvantages to a series of suggested visualization and interaction methods. The respondent's comments were then used to gauge the importance of these features, some of which are described below.

Suggestion: The provision of interactive system control

One of the forms of interaction suggested in the questionnaire was for the user to have control over the algorithm's execution and visualization. This is a common feature of many visualization systems which enables the user to monitor the effect of each program step (e.g. TPM [7], Balsa II [2], Tango [10]).

Typical Responses:

The respondents' comments associated with this type of control mechanism were favourable:

> "very useful - like an omniscient, but impotent viewer."

> "Excellent."

> "All of those options (bar one) are catered for in GAmeter, so I think they are useful! :) I know why you may want to step backward, but that's a lot of overhead on the GA."

Resulting Design: Visualization and Execution Control Panels

A visualization control panel could be used to flick back and forward through the algorithm's evolutionary history (figure 7). An execution control panel could provide the user with control over the algorithm's execution, the user could start and stop the algorithm's execution. A facility to save the algorithm's design, solution and/or evolutionary history would enable a record to be kept of each algorithm design. A facility to load a previously saved algorithm's data (either the algorithm's design, solution or history) would enable a new user to view previous users work and use this either as an educational aid, or as a model for their own algorithm design.

Figure 7 - An example of a visualization control panel. Icons for going back to the start of the search, jumping back ten generations, stepping back one generation, stepping forward one generation, jumping forward ten generations, and going to the end (respectively) are shown.

The figure above illustrates one example of a control panel; the visualization control panel, with which a user can; go back to the start of the algorithm's evolutionary search, jump back ten generations, step back one generation, step forward one generation, jump forward ten generations, or go forward to the end of the algorithm's evolutionary search (respectively).

Suggestion: A population fitness rating versus generation number graph

The survey questionnaire also asked users to comment on the suitability of a fitness rating versus generation graph (an example of which is shown in figure 8 below). This view illustrates the change in the populations' fitness rating across successive generations and is often used in GA texts to illustrate an algorithm's evolution (e.g. [8], [5], [1]).

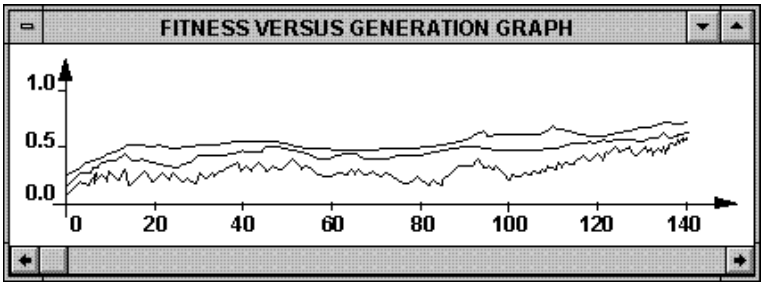

Figure 8 - An example of a fitness (y axis) versus generation number (x axis) graph illustrating the best fitness rating (top line), average fitness rating (middle line) and worst fitness rating (bottom line) per generation.

Typical Responses:

This was reported to be a useful illustration of an algorithm's rate of improvement. However, it was also noted that this isn't the whole picture:

> "Need more than this; need to know the local structure of fitness changes throughout the population."
>
> "Too much emphasis can be placed on the graph without going into any detail as to why that pattern occurred."
>
> "One often wants a more detailed understanding of what is happening in the population than this graph can give."

Although a three line graph containing the best, average and worst fitness ratings gives an impression of the range of the fitness ratings in each population, it gives no indication of the population diversity in terms of the number of solutions being considered.

Resulting Design: Complementary Views

Perhaps this view along with some form of population chromosome view (i.e. a view of the population's sample points in the search space, see figure 9) could be used to gain a more complete representation of an algorithm's evolution (e.g. [6], [3]). The strength of the fitness rating versus generation graph may be it's ability to highlighting important points in a solution's evolution which the user could then take a closer look at using a population chromosome view.

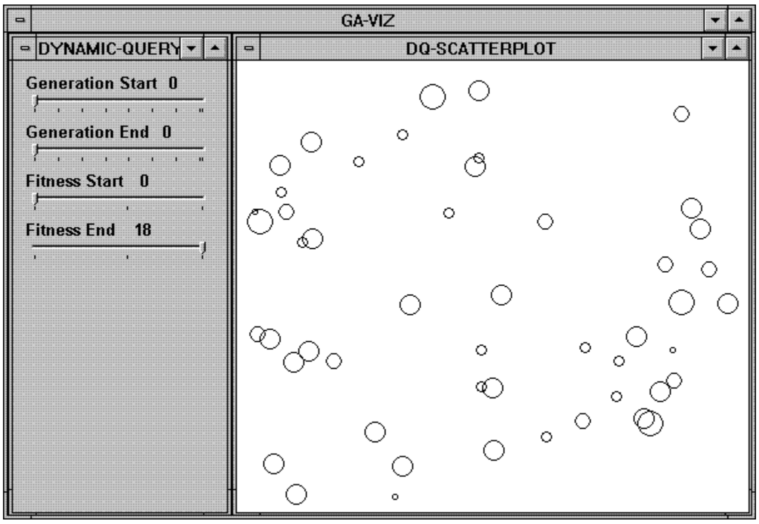

Figure 9 - An example population chromosome view showing the population's sample points in the search space.

An example of a population chromosome view is shown in figure 9 above. The circles on the right of the view show the sample points of the current generation (generation 0), the sliders to their left can be used to vary the range of generations and fitness ratings shown. The circles' diameter is used here to illustrate the individual's fitness ratings, the larger the diameter the larger the fitness. The scatter plot on the right uses a mapping of the chromosome binary strings into a two dimensional space in which similar chromosomes occupy similar regions of the space.

Requested views

In addition to asking users to describe their difficulties in designing GAs, and to comment upon some suggested visualization and interaction methods, the survey also asked the respondents for further ideas, things that the felt may make their work easier. One example of such an idea is discussed below.

An on-screen estimation of the time required to find a solution

One suggestion put forward by a respondent was to display an estimate of the time required to find an acceptable solution:

> "A good practical thing about making them easier to use -- assuming we're considering a typical industrial setting -- is an on-screen estimation, probably dynamic, on how long it will take to reach a given desired fitness. A large scale approximation based on fitness graph gradients would be fine."

Resulting Design: An Estimated Time to Solution View

An example of an on-screen estimation of the time required to find a solution is given in figure 10. This shows a prediction of the generation number at which an acceptable solution will be found and an estimated time to arrival based on multiplying the computer's average time to evolve a generation by the estimated number of generations required.

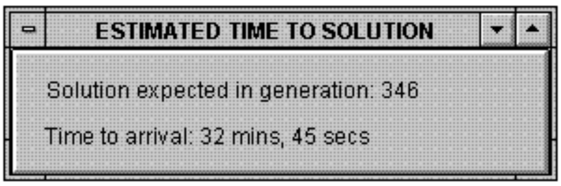

Figure 10 - An example of an "estimated time of arrival" view, including an estimate of the generation number at which a solution will be found and the time required to find the solution.

4 GA Visualization System Evaluation

This section shows an example screen view of a GA visualization system, and suggests two possible approaches that may be taken in order to evaluate such a system. Comments on these approaches are welcomed.

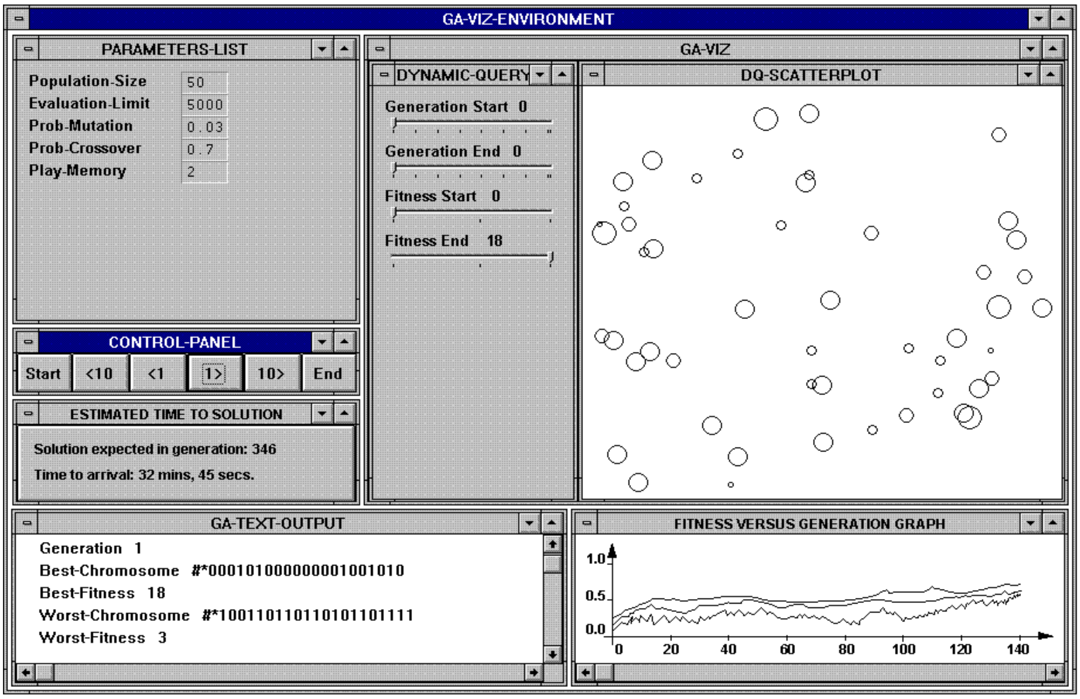

Figure 11 - An example of a possible screen view of the GA visualization system. This illustrates the; parameter dialogue box (top left), the user's visualization control panel (middle left), an estimated time to arrival view (middle left), a population chromosome view showing the current population's sample of the search space (top right), and a fitness versus generation number graph (bottom right). Each described in the previous section, as well as a text display showing the current population's best and worst individuals (bottom left).

In order to evaluate such a system an empirical study is to be undertaken. Although a variety of GA development environments exist there is no community standard environment to which this work could be compared. Instead users write their own software specifically for their use. This makes a comparative study difficult.

One initial plan for the evaluation is to make the software freely available on the condition that, after a few weeks of orientation with the system, users provide some feedback on its usefulness. This could be in the form of a questionnaire which asks the user to rate the system's support for the problem areas found in the initial user survey. However, without any common system to compare the GA visualization system to, this data may well be inconsistent across the sample group.

Another proposed evaluation method is to compare the use of the GA visualization system to the use of the user's own system, by asking the users first, to solve a problem using their own environment, and then (after a period of orientation) solve a second similar problem using the GA visualization system. The users' work would be recorded on video and a post-test interview could be conducted to gain further details on the user's opinions of their own system and the GA visualization system.

Finding an effective evaluation method is seen as a difficult problem but essential in order to validate this project. Any comments on the above evaluation approaches or alternatives would be welcomed.

References

[1] R. Belew and L. Booker, "Genetic Algorithms: Proceedings of the Fourth International Conference," : Morgan Kaufmann, 1991.

[2] M. H. Brown, *Algorithm Animation*. Cambridge, MA: MIT Press, 1987.

[3] T. D. Collins, "Genotypic-Space Mapping: Population Visualization for Genetic Algorithms," The Knowledge Media Institute, The Open University, Milton Keynes, UK, Technical Report KMI-TR-39, 30th September 1996.

[4] T. D. Collins, "The Visualization of Genetic Algorithms - Design Survey," presented at The Psychology of Programming Interest Group Postgraduate Student Workshop, Matlock, Derbyshire, UK, 1996.

[5] L. Davis, "Handbook of Genetic Algorithms," . New York: Van Nostrand Reinhold, 1991.

[6] R. Dybowski, T. D. Collins, and P. R. Weller, "Visualization of Binary String Convergence by Sammon Mapping," presented at The Fifth Annual Conference on Evolutionary Programming - EP96, San Diego, CA., 1996.

[7] M. Eisenstadt and M. Brayshaw, "The Transparent Prolog Machine (TPM): An Execution Model and Graphical Debugger for Logic Programming," The Open University, Human Cognition Research Laboratory, Technical Report 21A, October 1987.

[8] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*: Addison Wesley, 1989.

[9] B. A. Price, R. M. Baecker, and I. S. Small, "A Principled Taxonomy of Software Visualisation," *Journal of Visual Languages and Computing*, vol. 4, pp. 211 - 266, 1993.

[10] J. T. Stasko, "TANGO: A Framework and System for Algorithm Animation," : Brown University, 1989.

[11] D. Whitley, "A Genetic Algorithm Tutorial," Department of Computer Science, Colorado State University, Technical Report CS-93-103 (Revised), November 10th 1993.