

Open Research Online

The Open University's repository of research publications and other research outputs

The application of software visualization technology to evolutionary computation: a case study in Genetic Algorithms

Thesis

How to cite:

Collins, Trevor D. (1998). The application of software visualization technology to evolutionary computation: a case study in Genetic Algorithms. PhD thesis The Open University.

For guidance on citations see [FAQs](#).

© 1998 The Author

Version: Version of Record

Link(s) to article on publisher's website:

<http://people.kmi.open.ac.uk/trevor/archive/thesis/>

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data [policy](#) on reuse of materials please consult the policies page.

oro.open.ac.uk

**The Application of Software Visualization
Technology to Evolutionary Computation:
A Case Study in Genetic Algorithms**

A dissertation submitted in partial
satisfaction of the requirements for the degree
Doctor of Philosophy in Artificial Intelligence

Trevor D. Collins BEng. MSc.

KNOWLEDGE MEDIA INSTITUTE

The Open University, UK

Submitted - September 28, 1998

Acknowledgements

I am greatly indebted to my supervisor, John Domingue, for his advice and support. John has been an excellent supervisor providing insightful comments and constructive criticisms throughout this PhD project.

I would also like to thank my colleagues in the Knowledge Media Institute and the Psychology Department of The Open University for their advice, encouragement and friendship, without which I would certainly not have completed this thesis. In particular, I would like to thank David Bradbury, Richard Collenso, Shamus Foster, Tony Hirst, Simon Masterton, Enrico Motta, Paul Mulholland, Marco Ramoni, Tammy Sumner, Stuart Watt, Mike Wright and Zdenek Zdrahal for their active interest during the course of this project. A special word of appreciation goes to Shamus, Paul, Tammy and Stuart for their insightful comments on earlier drafts of this thesis. My gratitude also goes to Tom Routen for encouraging me to embark on this PhD.

I would like to take this opportunity to thank the study respondents who completed my questionnaire and the anonymous reviewers who provided constructive comments on the papers describing this work. I would also like to thank Richard Dybowski and Peter Weller for their help with the use of Sammon Mapping, Hartmut Pohlheim for his discussions on the visualization of a GA's search behaviour, and Annie Wu for her encouragement and interest in this project.

Finally, I would like to thank my family for their love, support and encouragement.

Related Publications

The following papers were published prior to the submission of this thesis and report some of the work contained herein.

Collins, T. D. (1998). Understanding evolutionary computing: A hands on approach. In D. B. Fogel (Ed.), *The Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98)*, part of the 1998 IEEE World Congress on Computational Intelligence (WCCI'98). Anchorage, Alaska: Morgan Kaufmann, San Mateo, CA. (May 1998). Awarded the ICEC'98 Best Student Paper Award.

Collins, T. D. (1997). Using software visualization technology to help evolutionary algorithm users validate their solutions. In T. Baeck (Ed.), *The Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*. East Lansing, MI: Morgan Kaufmann, San Mateo, CA. (August 1997).

Collins, T. D. (1997). Using software visualization technology to help genetic algorithm designers. In *The Proceedings of the Ninth Annual Conference of the Psychology of Programming Interest Group (PPIG'97)*, (pp. 43 - 51). Sheffield, UK. (January 1997).

Collins, T. D. (1996). Genotypic-space mapping: Population visualization for genetic algorithms, Technical Report No. KMi-TR-39b. Knowledge Media Institute, The Open University, Milton Keynes, UK.

Collins, T. D. (1996). The visualization of genetic algorithms - Design survey. In M. Ireland (Ed.), *The Proceedings of the First Psychology of Programming Interest Group Postgraduate Student*

Workshop, (pp. 35 - 52). Matlock, Derbyshire, UK: Group D Publications Limited, Sheffield, UK. (September 1996).

Dybowski, R., Collins, T. D., & Weller, P. R. (1996). Visualization of binary string convergence by Sammon mapping. In L. J. Fogel, P. J. Angeline, & T. Baeck (Ed.), *The Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, (pp. 377 - 383). San Diego, CA.: MIT Press. (August 1996).

Collins, T. D. (1995). The visualization of genetic algorithms - Related work, Technical Report No. KMi-TR-19. Knowledge Media Institute, The Open University, Milton Keynes, UK.

Abstract

Evolutionary computation is an area within the field of artificial intelligence that is founded upon the principles of biological evolution. Evolution can be defined as the process of gradual development. Evolutionary algorithms are typically applied as a generic problem solving method, searching a problem space in order to locate good solutions. These solutions are found through an iterative evolutionary search that progresses by means of gradual developments.

In the majority of cases of evolutionary computation the user is not aware of their algorithm's search behaviour. This causes two problems. First, the user has no way of assuring the quality of any solutions found other than to compare the solutions found by the algorithm with any available benchmark solutions or to re-run the algorithm and check if the results can be repeated or improved upon. Second, because the user is unaware of the algorithm's behaviour they have no way of identifying the contribution of the different components of the algorithm and therefore, no direct way of analyzing the algorithm's design and assigning credit to good algorithm components, or locating and improving ineffective algorithm components.

The artificial intelligence and engineering communities have been slow to accept evolutionary computation as a robust problem-solving method because, unlike case-based systems, rule-based systems or belief networks, they are unable to follow the algorithm's reasoning when locating a set of solutions in the problem space. During an evolutionary algorithm's execution the user may be able to see the results of the search but the search process itself like is a "black box" to the user. It is the search behaviour of evolutionary algorithms that needs to be understood by the user, in order for evolutionary computation to become more accepted within these communities.

The aim of software visualization is to help people understand and use computer software. Software visualization technology has been applied successfully to illustrate a variety of heuristic search

algorithms, programming languages and data structures. This thesis adopts software visualization as an approach for illustrating the search behaviour of evolutionary algorithms.

Genetic Algorithms (“GAs”) are used here as a specific case study to illustrate how software visualization may be applied to evolutionary computation. A set of visualization requirements are derived from the findings of a GA user study. A number of search space visualization techniques are examined for illustrating the search behaviour of a GA. “HENSON,” an extendable framework for developing visualization tools for genetic algorithms is presented. Finally, the application of the HENSON framework is illustrated by the development of “GONZO,” a visualization tool designed to enable GA users to explore their algorithm’s search behaviour.

The contributions made in this thesis extend into the areas of software visualization, evolutionary computation and the psychology of programming. The GA user study presented here is the first and only known study of the working practices of GA users. The search space visualization techniques proposed here have never been applied in this domain before, and the resulting interactive visualizations provide the GA user with a previously unavailable insight into their algorithm’s operation.

Contents

1	Introduction	33
1.1	Evolutionary Computation	34
1.2	Thesis Motivation	34
1.3	Software Visualization	35
1.4	Research Approach	36
1.5	Thesis Contributions	37
1.6	Thesis Overview	37
2	An Overview of Evolutionary Computation	39
2.1	Borrowing from Biology	40
2.1.1	EC Terminology	40
2.1.2	Natural Evolution	40
2.2	Evolutionary Algorithms	42
2.2.1	Evolutionary Programming	42
2.2.2	Evolution Strategies	44
2.2.3	Genetic Algorithms	46
2.3	Summary	49
3	GA User Study	52
3.1	Design	52
3.1.1	Issues	53
3.1.2	Delivery	54

3.1.3	Structure and Content	55
3.2	Results and Discussion	61
3.2.1	Background Information	61
3.2.2	Your Approach to GAs	62
3.2.3	What Characteristics to Visualize	68
3.2.4	Interaction Opportunities	74
3.2.5	Any Other Comments	76
3.2.6	Summary	76
3.3	Conclusions	80
4	Review of Related Work	83
4.1	Visualizing a GA's Key Characteristics	83
4.1.1	The Operation of the GA's Component Parts	84
4.1.2	The Quality of the Solutions Found by the GA	85
4.1.3	The Chromosomes' Genotypes	91
4.1.4	The Chromosomes' Phenotypes	98
4.1.5	The GA's Sampling of the Search Space	99
4.1.6	Navigating the GA's Search	103
4.1.7	Editing the GA	105
4.2	An Overview of the Existing Visualization Support	106
4.2.1	GA Systems	107
4.2.2	SV Systems	112
4.2.3	Information Visualization	124
4.3	Summary of the Contributions Made	126
5	Visualization Design Rationale	128
5.1	A Framework Approach Versus a Systems Approach	129
5.1.1	The Usability-Expressiveness Trade Off	130
5.1.2	Systems versus Frameworks	131
5.1.3	The Advantages of the Framework Approach	132

5.2	A Principled Approach to SV Design	133
5.2.1	Analysis of the Information	134
5.2.2	The Properties of the Graphic System	135
5.2.3	The Rules of the Graphic System	138
5.2.4	Design Summary	142
5.3	The Principled Design of Search Space Visualizations	143
5.3.1	Identifying Relevant Information	143
5.3.2	A Set of Applicable Graphical Schemata	144
5.3.3	A Set of Applicable Visualization Techniques	145
5.3.4	Discussion	155
5.3.5	Visualization Summary	157
6	HENSON: A GA Visualization Framework	160
6.1	Generic GA Players and Events	161
6.1.1	Generic GA Players	161
6.1.2	Generic GA Events	163
6.2	The HENSON GA View Hierarchy	167
6.3	The HENSON Architecture	169
6.4	Example GA Visualization Specifications	170
6.4.1	Fitness vs Time Graph	170
6.4.2	VIS - GA Visualization Tool	171
6.5	Summary	173
7	GONZO: A Search Space Visualization Tool	177
7.1	Design	177
7.1.1	Interface Design	179
7.1.2	HENSON Specification	185
7.2	Implementation	190
7.3	Application	196
7.3.1	Offline Visualization	196

7.3.2	Online Visualization	197
7.3.3	Interactive Command Line Control	198
7.4	Example Problem Visualizations	198
7.4.1	The Maximum Integer Problem	198
7.4.2	The De Jong F1 Test Problem	199
7.4.3	The Royal Road Problem	204
7.5	GUI Front End	206
7.5.1	GA Examples System Menu	207
7.5.2	View Specific Pop-Up Menu	207
7.6	User Walkthrough	209
7.6.1	GA Examples and Their Default Visualizations	209
7.6.2	Alternate GAs	210
7.6.3	Alternate Visualizations	211
7.7	Summary	211
8	Discussion	214
8.1	A Critique	215
8.1.1	The Validity of the Perceived Problem	215
8.1.2	Implementing a Principled Design Approach	216
8.1.3	The Usability of the Framework Approach	216
8.1.4	Really High Dimensional Visualizations	216
8.2	The Contributions of This Case Study	217
8.3	Future Work	219
8.3.1	Supporting GA design	220
8.3.2	Human-EA Interaction	225
8.3.3	Interactive Evolutionary Algorithms	226
8.3.4	More Search Space Representations	226
8.3.5	The Continued Development of HENSON	227
A	GA User Questionnaire	248

B GA User Study: Results Summary	259
B.1 Background Information	259
B.2 Your Approach to GAs	264
B.3 What Characteristics to Visualize	281
B.4 Interaction Opportunities	302
B.5 Any Other Comments	309
B.6 Future Contact	312
C GA User Questionnaire Responses	315
D GONZO Example Applications	449
D.1 Online Visualization	449
D.2 GONZO Example Problem Visualizations	450
D.2.1 The Maximum Integer Problem	450
D.2.2 The De Jong F1 Test Problem	452
D.2.3 The Royal Road Problem	455

List of Tables

2.1	A summary of the basic terminology used within EC.	41
2.2	An example pay-off matrix used to evaluate a weather predicting FSM. The <i>predicted</i> values are shown across the top, and the <i>actual</i> values along the left hand edge, the resulting pay-off is indicated at the intersection of the predicted and actual values. . .	44
2.3	An example of the state transitions, output and pay-off values for the FSM shown in Figure 2.2 reacting to the input sequence: sunny, sunny, rain, rain, rain, hurricane, hurricane, hurricane. The set of output values are displaced one position to the right in order to produce the predicted values, which are compared with the actual values (i.e. the input values) in order to identify the FSM's pay-off.	45
3.1	Question 1. A summary of the amount of time the respondents had been using GAs. .	61
3.2	Question 4. A tally of the environments used by the respondents. Showing the distribution of respondents using each machine, language and toolkit identified in the responses.	63
3.3	Question 5.1. A tally of comments made by the respondents with regard to the difficulties they encountered whilst defining the mapping between the problem domain and the string representation used by the GA. The respondents stated that was either a difficult task for them, not difficult for their particular problem, or important to the outcome of the GA. Note, <i>R4</i> considered this to be both difficult and important, hence the double entry.	64

3.4 Question 5.2. A tally of the comments made regarding the evaluation function’s definition. In addition to either being difficult or not difficult, some respondents noted that this was a difficult task only when there are conflicting criteria for evaluating the GA’s solutions to the problem, or that they were not interested in the development of the evaluation functions. 64

3.5 Question 5.3. A tally of the comments made regarding the selection of the GA’s components. As well as identifying this as either a difficult or not difficult task, some of the respondents considered this to follow directly from the representation they used and therefore not difficult, or they were unaware of the best components and used either default components that had worked in the past, followed existing guidelines published in the literature, or made modifications through unguided trial and error. 65

3.6 Question 5.4. A tally of the comments made regarding the selection of the GA’s parameters. The comments made here are similar to those made in Table 3.5. 66

3.7 Question 6.1. A tally of the steps carried out by the respondents to verify the quality of the solutions found by the GA. Specifically, respondents either did nothing more than check the fitness ratings of their solutions, verified their results by repeatedly running the algorithm, or compared the results of their algorithm to those of other alternative approaches. Note, respondent *A8* carried out both comparisons against the results of other algorithms and repeated runs. 66

3.8 Question 6.2. A tally of the steps carried put by the respondents to verify the quality of the GA’s search of the problem space. The respondents either did nothing to explore the quality of their GA’s search, or they examined the proposed solutions, compared the solutions given by alternative approaches, or in one case, explored the variation in fitness across the search space. 67

- 3.9 Question 7.1. The reported Advantages and Disadvantages of visualizing the individual chromosomes in the population. This was generally considered useful for seeing different aspects of the population, although some respondents considered this be of no advantage. The disadvantages noted here relate to the scalability of the visualization, in that it may present too much information and slow down the GA, or the information presented may confuse the user or may be difficult to represent. 68
- 3.10 Question 7.2. The Advantages and Disadvantages of visualizing a user defined selection of the chromosomes in the population. Again like Table 3.9, the advantages of viewing selected chromosomes relate to seeing what is happening in the population, this has the added advantage of being flexible to the user's requirements but also the disadvantage of perhaps under representing the chromosomes in the population and thereby confusing the user. 69
- 3.11 Question 7.3. The Advantages and Disadvantages of visualizing the rate of change in the populations' fitness ratings. The additional comments here relate to the user's interest in seeing more than just the chromosomes' fitness ratings, they also need to see how fitness relates to the local structure of the chromosomes. Concerns were also noted with regard to the effect visualization would have on the speed of the GA. . . . 70
- 3.12 Question 8.1. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool. Comments were with regard to validating the correct operation of the GA, also for problems in which the genotype is meaningless such views were considered to be not informative, not meaningful or not of interest to the user. 71
- 3.13 Question 8.2. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the occurrence of mutation in chromosomes. The comments made here were similar to previous comments, one of the respondents also considered this to the involve a high computational overhead for a relatively small contribution to their understanding. 72
- 3.14 Question 8.3. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the internal actions of the genetic operators. 73

3.15	Question 8.4. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing a "similarity" rating for each chromosome.	74
3.16	Question 10.1. The comments made by the respondents with regard to the use of a bi-directional execution control panel. The respondents could either do this already and did not comment further, considered this useful, in general or for educational or debugging purposes, or considered the use of navigational control more effective for offline visualizations than online visualizations.	75
3.17	Question 10.2. The comments made by the respondents with regard to using an editor to change the GA's parameters during execution. The comments here are similar to those in Table 3.16, two of the respondents however considered this to be disruptive to the GA's evolutionary search.	76
3.18	Question 10.3. The comments made by the respondents with regard to using an editor to change the population's chromosomes between generations. This was considered to be a strange idea that disrupts the GA's search and in some cases should not be attempted.	77
3.19	Question 10.4. The comments made by the respondents with regard to using an editor to change the gene-pool's chromosomes within a generation. The comments made here are similar to those referred to in Table 3.18.	78
4.1	The defining features of a range of EA fitness visualizations.	92
5.1	A taxonomy of sign-systems, each system is identified by the type of meaning attributed to the signs, and the system of perception used. This table was taken from [Bertin, 1983, page 2].	133
5.2	The levels of organization for visual variables. This table identifies the appropriate level of organization for each of Bertin's eight visual variables. These visual variables support either the perception of associations or dissociations, selective perception, the perception of order, and/or the perception of quantities. This table was taken from [Bertin, 1983, page 69].	135

5.3	A comparison of the computational complexity (expressed in big- O notation [Aho and Ullman, 1992]) of principal component analysis, Sammon mapping, Kohonen feature mapping and Bishop's latent-variable mapping. This table was taken from [Dybowski et al., 1996, page 382].	150
5.4	Sammon Mapping vs Extensive Repartitions: A comparison of the total error value for a complete search space mapping and the mean error value for each mapped datapoint is given here. The values listed for the Sammon mapping method are the average values taken from ten 100 iteration runs of the Sammon mapping algorithm.	151
5.5	A summary of the advantages and disadvantages of using Sammon Mapping and Search Space Matrices to visualize the search behaviour of a GA.	159
6.1	The HENSON GA specific players associated with a GA's principal class hierarchy. . .	162
6.2	The HENSON GA specific players associated with the details of the individual organisms. . .	163
6.3	The HENSON GA specific players associated with the statistical data of the GA's population.	164
6.4	The HENSON GA specific initialization events.	165
6.5	The HENSON GA specific evolution events.	166
6.6	The HENSON GA specific regenerate <i>selection</i> events used in the GA's evolution. . . .	167
6.7	The HENSON GA specific regenerate <i>reproduction</i> events used in the GA's evolution. .	168
6.8	The HENSON definition of the example fitness versus time graph given in Figure 6.4. .	172
6.9	The HENSON definition of the history, players, events and views used in the VIS GA visualization tool.	173
6.10	The HENSON definition of the <i>navigators</i> used in the VIS GA visualization tool. . . .	175
6.11	The HENSON definition of the <i>mappings</i> used in the VIS GA visualization tool. . . .	176
7.1	The HENSON definition of the <i>players</i> used and <i>events</i> recorded in GONZO.	186
7.2	The HENSON definition of the <i>views</i> available in GONZO.	187
7.3	The HENSON definition of the <i>mappings</i> used in GONZO.	188
7.4	The HENSON definition of the <i>navigators</i> available in GONZO.	190

B.1	Questions 1, 2 and 3. The areas of interest for each respondent in the theory group and the length of time they have been using GAs.	260
B.2	Questions 1,2 and 3. The areas of interest for each respondent in the research group and the length of time they have been using GAs.	261
B.3	Questions 1,2 and 3. The areas of interest for each respondent in the applications group and the length of time they have been using GAs.	262
B.4	Question 4. A summary of the environments used by members of the GA theory group	263
B.5	Question 4. A summary of the environments used by members of the GA research group	263
B.6	Question 4. A summary of the environments used by members of the GA applications group	264
B.7	Question 5.1. The difficulties found by members of the theory group whilst defining the mapping between the problem domain and the string representation used by the GA.	265
B.8	Question 5.1. The difficulties found by members of the research group whilst defining the mapping between the problem domain and the string representation used by the GA.	265
B.9	Question 5.1. The difficulties found by members of the applications group whilst defining the mapping between the problem domain and the string representation used by the GA.	266
B.10	Question 5.2. The difficulties found by members of the theory group whilst defining the evaluation function.	266
B.11	Question 5.2. The difficulties found by members of the research group whilst defining the evaluation function.	267
B.12	Question 5.2. The difficulties found by members of the applications group whilst defining the evaluation function.	268
B.13	Question 5.3. The difficulties found by members of the theory group whilst selecting the suitable algorithm components.	269
B.14	Question 5.3. The difficulties found by members of the research group whilst selecting the suitable algorithm components.	270

B.15 Question 5.3. The difficulties found by members of the applications group whilst selecting the suitable algorithm components.	271
B.16 Question 5.4. The difficulties found by members of the theory group whilst selecting suitable algorithm parameters.	272
B.17 Question 5.4. The difficulties found by members of the research group whilst selecting suitable algorithm parameters.	273
B.18 Question 5.4. The difficulties found by members of the applications group whilst selecting suitable algorithm parameters.	274
B.19 Question 5.5. The other set-up steps identified by members of the three user groups.	275
B.20 Question 6.1. The steps carried out by members of the theory group to verify the quality of their algorithm's solution(s).	276
B.21 Question 6.1. The steps carried out by members of the research group to verify the quality of their algorithm's solution(s).	277
B.22 Question 6.1. The steps carried out by members of the applications group to verify the quality of their algorithm's solution(s).	278
B.23 Question 6.2. The steps carried out by members of the theory group to verify the quality of their algorithm's search path.	278
B.24 Question 6.2. The steps carried out by members of the research group to verify the quality of their algorithm's search path.	279
B.25 Question 6.2. The steps carried out by members of the applications group to verify the quality of their algorithm's search path.	280
B.26 Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the theory group.	281
B.27 Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the research group.	282
B.28 Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the applications group.	283

B.29 Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the theory group.	284
B.30 Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the research group.	285
B.31 Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the applications group.	286
B.32 Question 7.3. The advantages and disadvantages of visualizing the rate of change in the populations' fitness values, as reported by members of the theory group.	287
B.33 Question 7.3. The advantages (Adv.) and disadvantages (Disadv.) of visualizing the rate of change in the populations' fitness values, as reported by members of the research group.	288
B.34 Question 7.3. The advantages and disadvantages of visualizing the rate of change in the populations' fitness values, as reported by members of the applications group.	289
B.35 Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the theory group.	290
B.36 Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the research group.	291
B.37 Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the applications group.	292
B.38 Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the theory group.	292
B.39 Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the research group.	293
B.40 Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the applications group.	294

B.41 Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the theory group.	294
B.42 Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the research group.	295
B.43 Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the applications group.	296
B.44 Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the theory group.	296
B.45 Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the research group.	297
B.46 Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the applications group.	298
B.47 Question 9. The additional features of a GA which members of the theory group reported that they would be interested in seeing.	299
B.48 Question 9. The additional features of a GA which members of the research group reported that they would be interested in seeing.	300
B.49 Question 9. The additional features of a GA which members of the applications group reported that they would be interested in seeing.	301
B.50 Question 10.1. The reported opinions of the members of the theory group on the use of a bi-directional execution control panel.	302
B.51 Question 10.1. The reported opinions of the members of the research group on the use of a bi-directional execution control panel.	303
B.52 Question 10.1. The reported opinions of the members of the applications group on the use of a bi-directional execution control panel.	304
B.53 Question 10.2. The reported opinions of the members of the theory group on the use of an editor to change their algorithm’s parameters during execution.	304
B.54 Question 10.2. The reported opinions of the members of the research group on the use of an editor to change their algorithm’s parameters during execution.	305

B.55	Question 10.2. The reported opinions of the members of the applications group on the use of an editor to change their algorithm's parameters during execution.	306
B.56	Question 10.3. The reported opinions of the members of the theory group on the use of an editor to alter the population's chromosomes between two generations.	306
B.57	Question 10.3. The reported opinions of the members of the research group on the use of an editor to alter the population's chromosomes between two generations.	307
B.58	Question 10.3. The reported opinions of the members of the applications group on the use of an editor to alter the population's chromosomes between two generations.	308
B.59	Question 10.4. The reported opinions of the members of the theory group on the use of an editor to alter the gene-pool's chromosomes within a generation.	309
B.60	Question 10.4. The reported opinions of the members of the research group on the use of an editor to alter the gene-pool's chromosomes within a generation.	310
B.61	Question 10.4. The reported opinions of the members of the applications group on the use of an editor to alter the gene-pool's chromosomes within a generation.	311
B.62	Question 11. The other forms of additional interaction reported by the members of the theory group.	311
B.63	Question 11. The other forms of additional interaction reported by the members of the research group.	312
B.64	Question 11. The other forms of additional interaction reported by the members of the applications group.	313
B.65	Question 12. All of the received additional suggestions for making GAs easier to use.	314

List of Figures

1.1	An overview of the research approach taken in this project.	36
2.1	Lewontin's Mappings. An illustration of the transformations involved in natural evolution; f_1 epigenesis, f_2 selection, f_3 genotypic survival, and f_4 mutation. This figure was taken from [Fogel, 1993, page 23].	42
2.2	An example of a weather predicting Finite State Machine ("FSM"). The network nodes indicate states and the links indicate input/output state transitions. The state values indicated here by s, R and H relate to sunny, Rain and Hurricane weather conditions. . .	43
2.3	An illustration of how "Roulette wheel selection" operates.	47
2.4	An illustration of how "Single point crossover" operates.	48
4.1	The internals window available in GIGA for showing the actions of the reproduction operators of a GA. This example was taken from [Dabs and Schoof, 1995, page 8]. . .	84
4.2	An XTANGO visualization illustrating a GA with a population containing three decimal valued chromosomes. The upper section illustrates the phenotype data i.e. the traveling salesman problem, the lower section shows an animated view of the genotype data (i.e. the decimal chromosomes) and the actions of the selection and reproduction operators used in the GA.	85
4.3	Two mean phenotype versus generation number graphs taken from [Fraser, 1957] (Figures 11 and 12).	86

4.4 An example of a 3D fitness graph. The fitness rating of each individual in the population is plotted over each generation. The fitness ratings are plotted on the y axis ($y = 0$ to 2.5), the position of each chromosome in the fitness ordered population is plotted on the x axis ($x = 0$ to 50), and the generation number is plotted on the z axis ($z = 0$ to 522). This figure was taken from [Harvey and Thompson, 1996]. 87

4.5 A Hinton style block diagram. This figure illustrates each chromosome in the population as a square block; the size of the block indicates the chromosome's fitness rating. Colour is used to highlight the chromosomes' level of fitness, here the fitness ratings are split into four bands corresponding to the four sizes of blocks used in the figure. This figure was taken from [Collins, 1993a] where texture was used to indicate colour on a black and white printer. 88

4.6 A Colour Map showing the fitness rating of every chromosome in a population, each chromosome is represented as a block; the block's colour indicates the chromosome's fitness rating. This figure was taken from [Collins, 1993a] where texture was used to indicate colour on a black and white printer. 88

4.7 A Coastline Fitness Diagram showing the chromosomes in two populations (a) and (b), here a fitness ordered view is shown on the left and a similarity (i.e. Hamming distance) ordered view is shown on the right for populations (a) and (b). Both views are ordered from left to right for increasing fitness and similarity ratings. This figure was taken from [Collins, 1993a]. 89

4.8 A radial plot of the fitness ratings in a single generation. The radial line trace shows the fitness ratings of the chromosomes in a fitness ordered population the distance (m) from the centre to the line indicates the magnitude of the fitness rating. 89

4.9 Three radial fitness plots illustrating three different stages during an algorithm's execution. 90

4.10	A radial fitness plot. Each individual's fitness rating is represented as a dot. The fitness ordered position of each chromosome is represented by the angular position of each dot, the distance (m) of a dot from the origin indicates the magnitude of the fitness rating, and the dot's colour indicates the generation number in which it last appeared.	90
4.11	Three example chromosome icons showing the design of line trace, DNA strip, and colour band icons. This figure was taken from [Collins, 1993a], where texture was used to indicate colour on a black and white printer.	93
4.12	A high dimensional visualization showing a population of 100, 1008 bit binary chromosomes as black and white pixels. The entire population is shown here in 100 rows each chromosome is shown as a single row of 1008 pixels.	94
4.13	An example of a VIS "run window," illustrating the best individual from each generation using a "zebra" representation.	94
4.14	An example of a VIS "population window" illustrating all the individuals in a single generation using a "zebra" representation.	95
4.15	An example of a VIS "individual window" showing the data held on a single individual.	96
4.16	Three example genotype visualizations; "overlaid chromosome icons" (left), a "population bar chart" (middle) and an "allele versus locus frequency matrix" (right). These three chromosome visualizations are taken from [Collins, 1993a].	96
4.17	Three example phenotype visualizations for the traveling salesperson problem. These images were taken from GIGA, XTANGO and EvoNet's Genetic Algorithm Software Development Package, respectively.	98
4.18	A 3D surface plot showing the fitness surface for a two dimensional search space. The chromosomes from old generations shown as blue dots and the chromosomes in the current generation shown as red dots, this figure was taken from [Spears, 1994].	100

- 4.19 Nassersharif, Ence and Au's scatterplot visualization for GAs solving two-dimensional problems. This figure (taken from [Nassersharif et al., 1994, page I-564]) shows scatterplots for generation 0 (left) and generation 10 (right). The x and z axes illustrate the two problem dimensions and the vertical y axis illustrates the fitness ratings, note the convergence toward fitter solutions shown in generation 10. 101
- 4.20 A data space view using the chromosomes fitness rating (y axis) and similarity to the fittest chromosome (x axis) to plot line trace icons of each chromosome. This figure was taken from [Collins, 1993a]. 102
- 4.21 The dialog boxes available in GAMETER for editing the GA's parameter (left) and algorithm settings (right). 105
- 4.22 The bit string editor dialog used in GAMETER to edit a chromosome's alleles. 105
- 4.23 A visualization of 2nd order schemata (i.e. schemata with two defined values). The four triangular images illustrate the frequency of four different 2nd order schema across all possible combinations of loci; 00 (top left), 01 (top right), 10 (bottom left), and 11 (bottom right). The frequency of each schema is indicated by the corresponding circle's gray value; a black circle indicates that the schema does not occur in the population through to a white circle which indicates the schema appears several times. 109
- 4.24 A visualization of the ancestry of a GA's population. The one hundred thirty bit chromosomes in the initial population were each given a separate colour, this visualization shows how the chromosomes from the initial generation have been recombined in order to produce the twenty fifth generation. 110
- 4.25 The main interface used in GIGA. Included in the interface are dialogs to alter the GA's parameters (top), a dialog to select views (bottom left), and a dialog to start, pause and step the GA's execution (bottom right). This figure was taken from [Dabs and Schoof, 1995, page 4]. 110
- 4.26 A screen image taken from Ronald Baecker's 30 minute movie on sorting algorithms. This image shows three insertion sort algorithms (left column), three exchange sort algorithms (middle column), and three selection sort algorithms (right column). 112

- 4.27 A screen shot taken from ZSTEP. In this code view the current focus of the stepper is shown in yellow and the recently substituted variable values are shown in red. 113
- 4.28 A screen shot taken from TPM version 1.11. The coarse-grained view at the top shows the complete execution space of the program, the individual AORTA diagrams at the bottom show a fine-grained view of the program's individual goals. 114
- 4.29 A pair of screen shots depicting the set-up phase of a Balsa session for a number sorting algorithm. The first screen view (left) illustrates the display layout selection dialogue in the centre of the screen. The second screen view (right) illustrates the parameter selection dialogue. In this particular example the user may select the initial organization of the numbers (currently set to a random ordering), the number of numbers to be sorted, and the random number generator's initial seed value. 115
- 4.30 A screen shot depicting the run phase of a Balsa session for a number sorting algorithm. The numbers are represented by vertical columns, the magnitude of each number is represented by the height of the corresponding column. As the numbers are sorted by the algorithm, the columns are moved and reordered by height. 116
- 4.31 John Stasko's algorithm animation framework as used in TANGO. This figure was taken from [Stasko, 1989, page 34]. 117
- 4.32 A screen view taken from a TANGO animation of a first-fit binpacking algorithm. The elements are inserted into the rectangle and tried against each column position until a large enough free-space is found to house them. The control bar shown at the bottom of the figure allows the user to pan around the view, zoom in and out, switch the debugger on/off, alter the refresh rate, and close the view. 118
- 4.33 The architecture of VIZ. This figure was taken from [Domingue et al., 1993, page 9]. . 119
- 4.34 An illustration of the software visualization support provided by VITAL. This figure was taken from [Domingue, 1995, page 8]. 120
- 4.35 A screen shot taken from a ZEUS binpacking algorithm animation. A control panel is shown in the top right window, a code visualization is shown in the bottom right window, an algorithm animation is shown in the bottom left window, and the algorithm's progress is shown in the top left data window. 121

- 4.36 An overview of PARADE highlighting its three major components; the “Parallel Program” component extracts the information required for producing the visualizations, the “Animation Choreographer” gathers the program information from the parallel program component and organizes it into a preferred format, and the “Visualization Paradigm” takes the choreographed program details and presents them in an apparently continuous smooth animation to the user. Any user interaction is passed to the animation choreographer by the visualization paradigm where it is acted upon. This figure was taken from [Stasko and Kramer, 1992, page 4]. 122
- 4.37 A hierarchy diagram illustrating the structure of a POLKA animation. The animator module controls the smooth animation of all the views by ensuring that each animation action is allocated a time-frame. This figure was taken from [Stasko and Kramer, 1992, page 5]. 123
- 4.38 Two screen shots showing 2D and 3D POLKA visualizations. The 2D visualization on the left shows the execution of a parallel quicksort algorithm, this visualization contains a control panel (top), a blocks view (left, height = value, horizontal position = position in array), and a chart view (right, vertical position = execution time, horizontal lines = swapping elements). The 3D visualization shown on the right shows the execution of a quicksort algorithm in a single 3D view (y axis = element value, x axis = execution time, z axis depth = position in array). 124
- 4.39 The FILMFINDER system which uses alphasliders to identify; film titles, leading actors, leading actresses, directors, and the film length. The x axis is used to indicate the year of release and the y axis indicates its popularity through cinema ticket sales. 125
- 5.1 The usability-expressiveness curve illustrating the trade off between usability and expressiveness as described in [Repenning and Ambach, 1996]. 130

- 5.2 An illustration of the levels of organization of each of Bertin's eight visual variables; the two planar dimensions, size, value, texture, colour, orientation and shape, as identified in Table 5.2. Included are examples for point, line and area implantations, the numbers shown on the left of the Selection column indicate the recommended number of levels to support selective perception by each implantation (point, line and area, respectively). This figure was taken from [Bertin, 1983, page 96]. 137
- 5.3 An illustration of Bertin's fifteen standard schema for constructing diagrams, networks and maps. This figure was taken from [Bertin, 1983, page 172]. 141
- 5.4 An illustration of Bertin's general rules of legibility, regarding graphic density (parts 1, 2, and 3), angular separation (4, 5, and 6), and retinal separation (parts 7,8, and 9). This figure was taken from [Bertin, 1983, page 174]. 142
- 5.5 The most efficient graphical schemata applicable for producing 2D and 3D visualizations of a GA's fitness landscape. 145
- 5.6 A Sammon map produced from a dataset of all the combinations of a 5 bit binary string. This mapping attempts to preserve the Euclidean distance between the points in the high (i.e. 5) dimensional search space and the two dimensional scatterplot representation. This is most effective for items around the edge of the 2D scatterplot. 147
- 5.7 A search space visualization of the progress of a GA one a problem with a single optimum solution. The chromosomes from generation 0, 2 and 20 are shown in the above three images, each circle illustrates a chromosome, the position of each circle illustrates the chromosome's location in a 2D Sammon map, the size of each circle illustrates the frequency of that chromosome in the population and the value image variable illustrates the chromosome's fitness rating. A line linking all the Hamming 1 neighbours of the fittest chromosome is shown in each image. 148
- 5.8 The construction of a complete search space matrix of a 4 bit binary dataset. The chromosomes at each position in the search space are shown on the right hand side, background shading is used to highlight the loci of the same alleles and italics are used to indicate the Hamming distance between neighbouring chromosomes. 149

- 5.9 A search space matrix illustrating the complete search space for a 4 bit coding alphabet of (0 1 2 3), (0 1 2), (0 1) and (0 1 2). 149
- 5.10 Two error surface plots for a Sammon mapping after 100 iterations from a random initial configuration (left) and an extensive repartitions mapping (right) of a six bit binary search space. 151
- 5.11 A point plot showing the best individual in a GA's population at a number of generations during the GA's run. Rather than mapping the entire search space, the Sammon map used here was produced by mapping only the chromosomes considered by the GA during its run. This figure was taken from [Pohlheim, 1998]. 152
- 5.12 A 6 bit binary search space matrix produced using a Gray coded weighting function. . 154
- 5.13 An example GONZO screen view illustrating how a search space visualization (top right) can be used to examine the structure of a specific chromosome 111111111111 (middle left). Each colour ribbon in the search space visualization (top right) identifies the regions of the search space that contain chromosomes with a 1 at the corresponding colour ribbon's locus, as shown in the Schema Seleccion dialog (middle left). The individual components of this screen view are explained in detail in Chapter 7. 155
- 5.14 A point plot showing all fifty individuals in a GA's population for every tenth generation of the GA's 5220 generation run. This view uses the first two principal components of a principal component analysis of all of the points to be shown, to plot each chromosome at a single 2D point. The initial population in this case appears as a scattering of points at the centre of the view (at 0,0) and slowly converges, whilst moving in a clockwise direction, to the cluster shown at the end of the trail to the left of the origin (at -5,0). 156
- 6.1 The inheritance hierarchy of views available in VIZ. This figure was taken from [Domingue et al., 1993, page 12]. 169
- 6.2 The HENSON view hierarchy. The extensions to the VIZ view hierarchy are shown here in bold boxes with italicised labels. 170

- 6.3 The architecture of the HENSON GA visualization framework. The four main modules; History, View, Mapping and Navigators, are shown in rectangular boxes, as in the VIZ architecture diagram. However, HENSON links the navigator module to the visualization source module in order to enable the user's editing of the GA's parameters or components. 171
- 6.4 An example GA visualization, a fitness versus time graph showing the minimum (blue line), average (green line) and maximum (red line) fitness ratings in each population for a 29 generation GA run (0 to 28). 171
- 7.1 An example screen image taken from GONZO. This example includes three views; a coarse-grained fitness versus time graph (bottom right), a medium-grained search space visualization (top right) and a fine-grained chromosome view (bottom left), and three navigators; a movie control panel (top left), generation and fitness range selector (second left), and a schema highlight selector (third left). The search space visualization currently illustrates all the chromosomes considered by the GA between generations 11 and 17 with fitness ratings between 3050 and 4095. The GA is attempting to solve the 12 bit maximum integer problem. This problem seeks out binary chromosomes with high integer values. 180
- 7.2 An example of the fitness versus time graph available in GONZO. The top red line trace indicates the best fitness rating in each population, the bottom blue line trace indicates the worst fitness rating in each population, and the middle green line trace indicates the average value of all the chromosomes' fitness ratings. A superimposed grey rectangle indicates the region of the GA's run that is currently being displayed in the search space visualization (i.e. the chromosomes considered between generation 11 and generation 17 with fitness ratings between 3050 and 4095). 181

7.3 An example of a search space visualization. The search space visualization is shown here on the left with an enlarged section of its schema legend shown on the right, the string highlighted here is 111111111111. Each chromosome is indicated as a point, at each point a circle is drawn, where the size of the circle indicates the chromosome’s fitness. The circles range from small circles for low fitness values to large circles for a high fitness values. 182

7.4 An example of a fine-grained chromosome view displaying the chromosome value and fitness rating of eight selected chromosomes. 183

7.5 The movie player control panel used in GONZO to navigate the GA’s execution. 183

7.6 The alphasliders range selector used in GONZO to define a range of generations and fitness ratings to be displayed in the search space view. 184

7.7 The schema highlighting dialog used in GONZO. The location of the chromosomes containing the schema identified by the button labels are highlighted using the colour coded ribbons in the legend of the search space matrix. Clicking on each button makes the label change to the next allele in the GA’s coding alphabet. 185

7.8 The architecture of GONZO. Here the distinction between the GECO GA prototyping environment and GONZO visualization tool is made explicit along with the content and direction of communication made between each module. Dashed lines are used to distinguish the information required to initialize GONZO. 191

7.9 Four screen images taken from GONZO showing the population of a GA solving the 12 bit Maximum Integer problem after generations 0, 9, 18 and 28 (top left, top right, bottom left and bottom right, respectively). 199

7.10 A screen image taken from GONZO showing the complete population data of a GA solving the 12 bit Maximum Integer problem. The size of each point shown on in the scatterplot view (top left) indicates the magnitude of the fitness rating for the chromosome at that position in the search space matrix. In this case the magnitude of the fitness ratings ranges from 0 at the bottom left hand corner of the search space matrix to 4095 ($2^{12} - 1$) at the top right hand corner of the search space matrix. . . . 200

7.11 A 3D illustration of part of the fitness landscape of De Jong’s F1 test problem. The decimal values of two of the three problem dimensions are plotted here on the x and z axes and the corresponding fitness rating is shown on the y axes of the 3D surface plot. This problem attempts to minimize the fitness rating such that the GA evolves solutions located around the centre of the above fitness surface. 201

7.12 A series of example GONZO screen images for a GA solving De Jong’s F1 test problem. The same screen layout is used here as in Figure 7.9, the state of the GA at generation 0 (top left), 20 (top right), 40 (middle left), 60 (middle right), and 81 (bottom left) are shown along with the complete GA’s run (generations 0 to 81, bottom right). In this example each chromosome is illustrated by a rectangle, the colour of each rectangle indicates the corresponding chromosome’s fitness rating. The colours range from red for a high (i.e. poor) fitness rating, to blue for a low (i.e. good) fitness rating. The projection order for the search space mapping is taken from the most significant bit for each problem dimension i.e. (0 11 21 1 12 22 2 ... 28 9 19 29). 202

7.13 A set of fitness bound GONZO screen images for a GA solving De Jong’s F1 test problem. Three fitness ranges are shown here, showing chromosomes within the ranges 26 to 35 (top left), 26 to 30 (top right) and 26 to 27 (bottom left and right). The bottom two screen images show two different schema selected by the user in the schema highlighting dialog (third left). These illustrate the different schema structures held by two near optimum solutions (shown in these two screen images as the lowest right-most coloured box and the highest left-most coloured box in the search space matrix). The same screen layout is used here as in Figure 7.12. 203

7.14 An example screen image taken from GONZO for viewing a single run of a GA solving the Royal Road function [Mitchell et al., 1991]. Eight search space visualizations are used here to illustrate the eight sections of the sixty four bit chromosomes. The four scatterplots at the top of the screen view represent the first four building blocks for loci 0 to 7, 7 to 15, 16 to 23 and 24 to 31, and the four scatterplots in the middle of the screen view represent the last four building blocks for loci 32 to 39, 40 to 47, 48 to 55 and 56 to 63. 205

- 7.15 The system menu bar used in GONZO to select example GAs. Three options are available in this menu, these run the GA and present the visualizations for the maximum integer problem, De Jong’s F1 test problem, and the royal road problem (as described in Section 7.4). 207
- 7.16 The pop-up menu bar used in GONZO to identify the image mapping used in the search space visualization. Five options are available in this menu for circle size, box size, box colour, box darkness and box lightness image mappings. 208
- 7.17 Getting started with GONZO. This figure shows the three stages involved in executing GONZO: (1) loading the gonzo.lsp file that creates GONZO, (2) starting GONZO with the command “(gonzo)” in the Lisp listener’s command line, and (3) selecting an example GA application from the GONZO system menu. 209

Chapter 1

Introduction

Evolutionary Computation (“EC”) is the study of computing techniques based on the guiding evolutionary principle of “survival of the fittest.” Evolutionary Algorithms (“EAs”) are powerful generic search algorithms capable of finding good solutions to complex problems. Some example areas in which EAs have been applied for problem solving and modeling include; optimization, automatic programming, machine learning, economics, immune systems, ecology, population genetics, evolution and learning, and social systems (see [Goldberg, 1989], [Ross and Corne, 1994], [Alander, 1995] and [Mitchell, 1996] for examples).

The problem with EC is that people find it difficult to understand the evolutionary search behaviour of their algorithms. Although searching the problem space by simulated evolution biases the search toward the better regions of the problem space, hundreds, if not thousands, of solutions are considered during a typical EA’s execution. Summary statistics can be used to give an impression of the algorithm’s evolution, such as the best, average and worst quality of the solutions contained in each population. However, at the beginning of this project there were no methods capable of supporting the EA user’s comprehensive understanding of their algorithms’ evolutionary search behaviour.

The primary objective of Software Visualization (“SV”) is to facilitate peoples’ understanding and effective use of computer software [Price et al., 1993]. This has been used successfully to illustrate the operation of programming languages [Eisenstadt and Brayshaw, 1987], [Reiss, 1990], [Lieberman and Fry, 1995], computer algorithms [Brown and Sedgewick, 1985], [Stasko, 1989], [Brown, 1991], and the effects of a program on a dataset [Moher, 1988], [Roman et al., 1992]. This

this thesis explores how SV technology may be applied to support EC. This chapter introduces the two main themes of this work; Evolutionary Computation and Software Visualization, and explains the motivation, research approach, contributions and structure of this thesis.

1.1 Evolutionary Computation

Evolutionary Computation is a rapidly expanding area of artificial intelligence research, with more than twenty international events per year and at least half a dozen journals, over a thousand EC related papers are published per year [Schwefel and Kursawe, 1998].

Within EC there are three classes of EA; Evolutionary Programming, Evolution Strategies, and Genetic Algorithms. These classifications are based on the level in the hierarchy of evolution being modeled by the algorithm. Evolutionary Programming (“EP”) models evolution as a process of adaptive species. Evolution Strategies (“ESs”) models evolution as a process of the adaptive behaviour of individuals. Thirdly, Genetic Algorithms (“GAs”) models evolution at the level of genetic chromosomes i.e. the basic instructions for making things.

EAs do not necessarily locate the optimal solution to a problem, the advantage of EAs is that they find “acceptably good” solutions to problems “acceptably quickly” [Beasley et al., 1993]. In their overview of GAs Beasley, Bull and Martin note that “where specialized techniques exist for solving particular problems, they are likely to out-perform GAs in both speed and accuracy of the final result” [Beasley et al., 1993, page 58]. It is in difficult areas where no such techniques exist that EAs should be applied. In these areas, the size of the problem space is such that an exhaustive search is impractical, and the structure of the problem space is such that traditional search algorithms are ineffective. EAs excel by striking a balance between the continued exploration of the problem space and the exploitation of the useful components held in the solutions discovered so far.

1.2 Thesis Motivation

The problem with EC is that EAs search large problem spaces by making gradual improvements to a set of possible solutions. There is no single point during the algorithm’s run that can be held responsible for the outcome, the solutions emerge during the course of the algorithm’s iterations.

This results in a fundamental credit assignment problem for EA users i.e. if good solutions are found what proportion of the credit should be attributed to the individual components of the algorithm's design?

This problem is further compounded by the fact that the users are unable to see the EA's search behaviour. EA users commonly examine how the quality of the solutions found by their algorithm changes over time using a graph of the population's fitness versus generation number. Although this graph illustrates the improvements in the quality of the solutions considered during the algorithm's run, it does not illustrate anything about the structure of the solutions being considered, or the regions of the search space being explored.

The aim of this project is to address this fundamental design problem by applying software visualization techniques to enable the user to examine the structure of the solutions being considered and the regions of the search space being explored. By enabling the user to see the search behaviour of their algorithms, they can then begin to attribute credit to the individual designs and judge the quality of each algorithm based on its exploration of the problem space.

1.3 Software Visualization

Software Visualization ("SV") has been defined as "the use of the crafts of typography, graphic design, animation and cinematography with modern human-computer interaction technology to facilitate the human understanding and effective use of computer software" [Price et al., 1993]. Visualization is specifically intended to enable the user to interact with, as well as observe, their data [McCormick et al., 1987]. A recent empirical evaluation of SV found that students who were able to control and interact with a variety of algorithm animations gained a better understanding of the algorithms' behaviour than those who could only passively observe the visualizations [Lawrence et al., 1994].

The application of visualization techniques to support peoples understanding of EAs has been receiving growing attention during the last few years; [Kapsalis and Smith, 1992], [Routen and Collins, 1993], [Chipperfield et al., 1994], [Nassersharif et al., 1994], [Dabs and Schoof, 1995], [Dybowski et al., 1996], [Harvey and Thompson, 1996], [Collins, 1997]

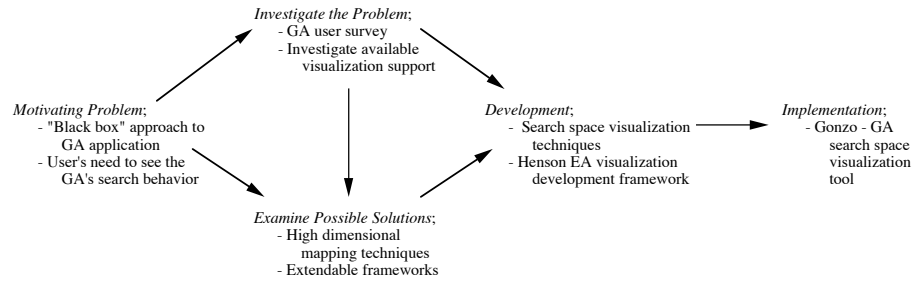


Figure 1.1: An overview of the research approach taken in this project.

and [Shine and Eick, 1997]. By enabling EC users to observe and interact with EAs it is hoped that a better understanding of their behaviour will be achieved.

1.4 Research Approach

This section describes the research approach taken in this project. There were essentially five stages in this project. Figure 1.1 illustrates each stage; from the initial *problem description*, through the *investigation* of the problem, the *examination* of some possible solutions, and the *development* of specific solutions, to the *implementation* of an example (proof of concept) visualization tool.

At the start of this project the decision was taken to examine GAs as a specific case study of EC visualization. A case study approach was considered important for this project, in order to identify the unique visualization requirements of a specific EA, as well as the generic visualization requirements of EC. Although all three types of EA are based on a common metaphor, the slight differences between their use of that metaphor results in significant differences in their implementation and application. As will be seen in the following chapter, GAs are significantly different to both ESs and EP, in that they emphasize genotypic rather than phenotypic transformations, i.e. the level in the evolutionary hierarchy at which GAs operate is very different to that of ESs or EP. However, the purpose of this thesis is to explore how visualization technology may be best applied to EC. Therefore, it is the generic approach to the provision of SV support for EC and the identification of generic EC, and specific GA, visualization techniques that are of importance. GA visualization was chosen as the specific EC domain in order to build on the existing body of work, see Section 4.1.

The motivating problem with GAs (as described in Section 1.2), is that GAs are difficult to apply because the user currently has no way of seeing the GA's exploration of the problem space. The

validity, importance and consequences of this problem were investigated through a GA user study. The study questionnaire was completed by nineteen GA users with a diverse range of reasons for using GAs. The responses to the questionnaire were used to establish a set of GA users' visualization requirements, and the contributions made by the available visualization support for fulfilling these requirements was examined.

Based on these results, a proposal was made to develop search space visualizations to support a user's understanding of the GA's search behaviour, and an extendable GA visualization framework with which GA users could develop their own visualizations. The development of the search space visualizations involved the investigation of a number of multivariate scaling techniques to produce two dimensional representations of the chromosomes in the GA's high dimensional search space. The "HENSON" visualization framework was produced to support the development of GA visualization tools. Finally, an example GA search space visualization tool called "GONZO" was implemented using HENSON.

1.5 Thesis Contributions

This thesis makes the following contributions:

- Software Visualization
 1. "HENSON," a framework for developing GA visualization tools.
- Evolutionary Computation
 1. A set of GA users' visualization requirements.
 2. The development of a set of high dimensional search space visualizations suitable for exploring a GA's search path.
 3. "GONZO," a GA visualization tool for exploring the evolutionary search behaviour of GAs.

1.6 Thesis Overview

The objective of this project is to examine how SV techniques may be most effectively applied to support peoples understanding and use of EC. A visualization framework has been developed and

applied to produce a set of GA visualizations as a case study in EC visualization.

An overview of EC is given in Chapter 2. Chapter 3 presents the findings of a study carried out in order to identify the working practices of GA users, the difficulties they experience while applying GAs and their opinions regarding the potential use of visualization. The results of the study identify a set of GA users' visualization requirements. The degree of support available from the existing work done on visualizing GAs is then explored in Chapter 4 along with some relevant contributions from the SV, information visualization, and human-computer interaction communities.

Chapter 5 discusses the design rationale used in this project. Specifically, it looks at the graphic design principles used, the advantages of using an extendable framework for developing GA visualization tools over a closed GA visualization system, and a series of visualization techniques for presenting the search path of a GA. Chapter 6 introduces "HENSON," a framework for developing GA visualization tools. Chapter 7 presents "GONZO," an example GA visualization tool implemented using the HENSON framework, that supports the user's perception of a GA's search path. Chapter 8 critiques some of the work presented here and concludes this thesis with a summary of the contributions made and a speculative discussion of future work.

Chapter 2

An Overview of Evolutionary Computation

This chapter gives an overview of EC in order to establish the background context for the visualization work carried out in this thesis. For a more complete understanding of the field of EC see [Goldberg, 1989], [Beasley et al., 1993], [Fogel, 1993], [Schwefel, 1995], [Baeck, 1996], [Hand, 1994], or [Baeck et al., 1997].

The precise origin of EC is difficult to define, a number of authors are commonly cited for originating EC, including [Anderson, 1953], [Fraser, 1957], [Friedberg, 1958], [Ashby, 1960] and [Bremermann, 1962] (see [Fogel, 1998a] for a more complete discussion). Three different forms of EC are recognized today: Evolutionary Programming (“EP”), Evolution Strategies (“ESs”), and Genetic Algorithms (“GAs”). Each was developed independently during the 1960s and early 1970s:

- EPs were proposed by Lawrence Fogel, Alvin Owens and Michael Walsh whilst examining the use of simulated evolution as an approach for developing artificial intelligence [Fogel et al., 1966].
- ESs were created by Ingo Rechenberg and Hans-Paul Schwefel at the Technical University of Berlin as experimental optimum-parameter seeking procedures and numerical optimization algorithms [Rechenberg, 1973], [Schwefel, 1995].
- GAs were introduced by John Holland at the University of Michigan whilst working on the use

of evolutionary techniques for adaptive systems [Holland, 1970], [Holland, 1975].

Although developed independently, these were all inspired by the principles of evolution. This thesis examines GA visualization as an example of EC visualization. The application of the visualization approach and techniques developed here for GAs to other forms of EC is discussed in Chapter 8.

A basic introductory overview of the key biological concepts and terminology is presented in Section 2.1. Each of the three forms of simulated evolution; EP, ESs and GAs, are then described in Section 2.2. Finally, Section 2.3 discusses the differentiating features of EP, ESs and GAs, and highlights the key characteristics of GAs that may play an important role in understanding their search behaviour.

2.1 Borrowing from Biology

EC is based on the principles of biological evolution. In order to explain the formulation of each of the three different forms of EC, this section gives a basic introduction to some of the terminology used by the EC community and a brief overview of the key concepts of natural evolution upon which EAs are based.

2.1.1 EC Terminology

Table 2.1 presents a brief overview of some of the terminology borrowed from biology and used in EC. A fuller description of the technical terms used in this thesis is available in the Glossary.

2.1.2 Natural Evolution

Evolution is the product of gradual development [Hawkins and Allen, 1991]. Living organisms evolve through the interaction of competition, selection, reproduction and mutation processes. The evolution of a population of organisms can be described using “Lewontin’s mappings” [Lewontin, 1974]. These highlight the differences between an organism’s “genotype” and “phenotype.” The *genotype* is the organism’s underlying genetic coding (DNA) . The *phenotype* is the manner of response contained in the physiology, morphology and behaviour of the organism (see also [Fogel, 1993]) .

Table 2.1: A summary of the basic terminology used within EC.

BIOLOGICAL TERM	EC MEANING
chromosome	string of symbols
population	a set of chromosomes
deme	a local population of closely related chromosomes, a subset of the total population
gene	a feature, character, or detector
allele	feature value
locus	a position in a chromosome
genotype	structure
phenotype	a set of parameters, an alternative solution, or a decoded structure

Figure 2.1 on page 42 illustrates the four subprocesses of evolution; “epigenesis,” “selection,” “genotypic survival” and “mutation.” The function f_1 , *epigenesis*, maps the population of genotypes, $g_1 \in \mathbf{G}$, to the phenotypic state space, \mathbf{P} as a set of phenotypic expressions, p_1 . The result is partially dependent on the environment, which can be expressed as a set of symbols, $(i_1, \dots, i_k) \in \mathbf{I}$, where \mathbf{I} is the set of all such environment sequences.

Function f_2 , *selection*, maps the set of phenotypic expressions p_1 into p_2 . Note, selection operates only on the phenotypic expressions of the genotype, the underlying coding g_1 is not involved in the selection process. The function f_3 , *genotypic survival*, describes the effects of the selection and migration processes which occurred under f_2 , on \mathbf{G} . Finally, function f_4 , *mutation*, maps the population of genotypes $g_2 \in \mathbf{G}$ to $g_1' \in \mathbf{G}$. This function represents the operation of mutation (including any recombination and higher level mutations).

These four subprocesses combined map a population, $g_1 \in \mathbf{G}$, to $g_1' \in \mathbf{G}$. Evolution (i.e. gradual development) occurs over the successive iterations of these mappings.

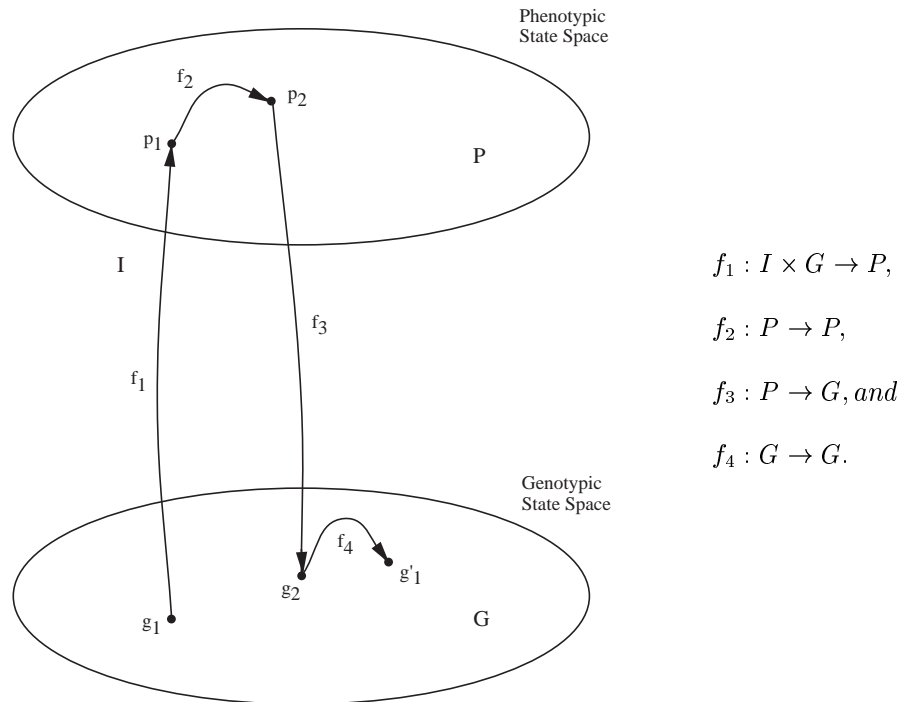


Figure 2.1: Lewontin's Mappings. An illustration of the transformations involved in natural evolution; f_1 epigenesis, f_2 selection, f_3 genotypic survival, and f_4 mutation. This figure was taken from [Fogel, 1993, page 23].

2.2 Evolutionary Algorithms

Within this thesis the term “evolutionary algorithm” is used to refer to the all forms of software-based evolutionary computation (evolutionary hardware is not considered here). The following three subsections briefly describe the three main forms of EC; EP, ESs and GAs. Although a complete review of EC is beyond the scope of this thesis an overview of each type of EA is given with an emphasis on showing how these three types of algorithm differ and the stages involved in defining each one. The aim of this overview is to enable the clear differentiation of each type of EA.

2.2.1 Evolutionary Programming

Evolutionary Programming (“EP”) emphasizes phenotypic adaption, i.e. they emphasize the behavioural link between the parent chromosomes and their offspring. Each chromosome identifies the behaviour (i.e. phenotypic traits) of a Finite State Machine (“FSM”). An FSM is a “machine defined in terms of a finite alphabet of possible input symbols, a finite alphabet of possible output symbols, and some finite number of possible different internal states” [Fogel et al., 1966, page 12]. Each state

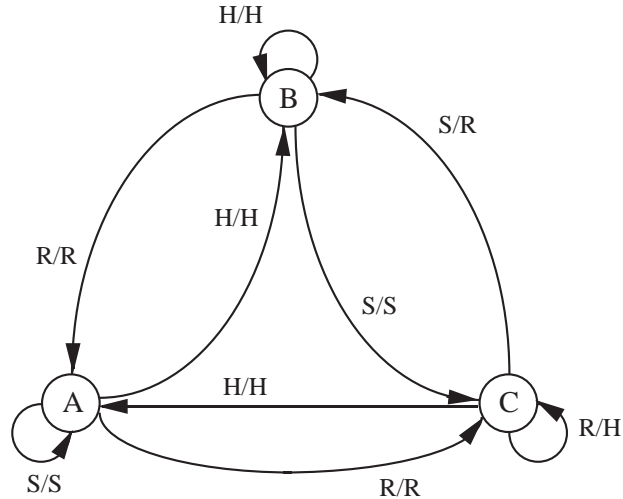


Figure 2.2: An example of a weather predicting Finite State Machine (“FSM”). The network nodes indicate states and the links indicate input/output state transitions. The state values indicated here by s, r and h relate to sunny, rain and hurricane weather conditions.

is indicated as a node in a network and the network links indicate the input/output state transitions. An example of a weather predicting FSM is given in Figure 2.2. These FSMs can be represented as a list of start node, end node, input value, output value quadruples. For example the circular link on the left of Figure 2.2, linking node A to node A with the input value sunny and output value sunny (labeled “s/s”), can be represented by A A S S. Adopting this representation protocol the FSM shown in Figure 2.2 could be represented as follows; A A S S A B H H A C R R B B H H B A R R B C S S C C R H C B S R C A H H, where A, B and C are node names, and s, r and h correspond to the state values sunny, Rain and Hurricane.

An FSM starts in a specific state (e.g. node A) and acts according to a supplied input pattern. The quality of an FSM’s predictions can be judged using a pay-off matrix (see Table 2.2). An example set of inputs and the resulting predictions of the FSM shown in Figure 2.2 is given in Table 2.3. The set of output symbols are moved one position to the right in the table in order to compare the predicted values (outputs) to the actual values (inputs). The predictive ability of the resulting FSM is found by summing the pay-off values from each prediction and dividing the result by the number of predictions made. For the input sequence S S R R R H H H the example FSM shown in Figure 2.2 has a predictive ability of 40.14 (+281 ÷ 7).

A typical EP works as follows: An initial population is created using a random number generator

Table 2.2: An example pay-off matrix used to evaluate a weather predicting FSM. The *predicted* values are shown across the top, and the *actual* values along the left hand edge, the resulting pay-off is indicated at the intersection of the predicted and actual values.

ACTUAL VALUES	PREDICTED VALUES		
	<i>Sunny</i>	<i>Rain</i>	<i>Hurricane</i>
<i>Sunny</i>	+1	-5	-50
<i>Rain</i>	-20	+10	-10
<i>Hurricane</i>	-100	-75	+100

and evaluated using the problem specific pay-off function. Each chromosome is then mutated to create a new population of offspring. Mutation involves either; the addition, deletion, change of output, change of transition of a node, or a change of starting node. The offspring are then evaluated and the better half of the combined set of parents and offspring is used as the next population. This evolutionary process is repeated until an acceptable solution is found.

Note, a chromosome in EP encodes the behaviour of an individual (i.e. its phenotypic state). Mutation is the only mating operator that is applied and it is applied to every individual irrespective of their evaluated pay-off. Selection is made from the combined set of parents *plus* offspring.

Application

In order to define an EP the user must complete the following two tasks:

1. Define the initial sequence of symbols as the observed environment, and
2. Define the pay-off function.

2.2.2 Evolution Strategies

Evolution Strategies (“ESs”), like EPs, emphasize phenotypic transformations. Hans-Paul Schwefel worked on the first computer implementations of ESs in the early 1970s [Schwefel, 1975]. Between the years of 1976 and 1985 little more work was done on ESs, due to a lack of financial support. However, during the last decade academic interest as well as financial support for ESs have been

Table 2.3: An example of the state transitions, output and pay-off values for the FSM shown in Figure 2.2 reacting to the input sequence: sunny, sunny, rain, rain, rain, hurricane, hurricane, hurricane. The set of output values are displaced one position to the right in order to produce the predicted values, which are compared with the actual values (i.e. the input values) in order to identify the FSM's pay-off.

<i>Present State</i>	A	A	A	C	C	C	A	B
<i>Input Value</i>	Sunny	Sunny	Rain	Rain	Rain	Hurricane	Hurricane	Hurricane
<i>Next State</i>	A	A	C	C	C	A	B	B
<i>Output Value</i>	Sunny	Sunny	Rain	Hurricane	Hurricane	Hurricane	Hurricane	Hurricane
<i>Predicted Value</i>		Sunny	Sunny	Rain	Hurricane	Hurricane	Hurricane	Hurricane
<i>Actual Value</i>	Sunny	Sunny	Rain	Rain	Rain	Hurricane	Hurricane	Hurricane
<i>Pay-off</i>		+1	-20	+10	-10	+100	+100	+100

revived [Schwefel, 1995].

The original application of these strategies dealt with the optimization of hydrodynamical problems [Baek and Hoffmeister, 1994], such as optimizing the shape of a bent pipe [Lichtfuss, 1965], minimizing the drag of a joint plate [Rechenberg, 1965], and optimizing the structure of a flashing nozzle [Schwefel, 1968]. These ESs evolve by making a series of discrete adjustments (i.e. mutations) to an experimental structure. After each adjustment, the new structure, i.e. the off-spring, is evaluated and compared to the previous structure, i.e. the parent. The better of the two is then chosen and used in the next cycle. As selection in this evolutionary cycle is made from one parent *and* one off-spring, the algorithm is known as a “(1 + 1)” ES.

These *two-membered* ESs modify (i.e. mutate) an n -dimensional real-valued vector $\mathbf{x} \in \mathbb{R}^n$ of object variables by adding a normally distributed random variable with expectation zero and standard deviation σ to each of the object variables x_i . The standard deviation is the same for all components of \mathbf{x} , i.e. $\forall i \in \{1, \dots, n\} : x_i' = x_i + \sigma N_i(0, 1)$, where x' is the off-spring of x and $N_i(0, 1)$ is the realization of a normally distributed random variable with expectation 0 and standard deviation 1.

Since the introduction of ESs, two additional strategies have been developed: $(\mu + \lambda)$ and (μ, λ) . Both of these ESs work on populations rather than single individuals and are referred to as *multi-*

membered ESs. A $(\mu + \lambda)$ ES creates λ off-spring from μ parents and selects the best μ individuals from the combined set of μ parents *plus* λ off-spring to make the next population. A (μ, λ) ES, on the other hand, creates λ off-spring and selects the best μ individuals from the off-spring alone. In general $(1 \leq \mu \leq \lambda)$.

Application

In order to define an ES the user must complete the following two tasks:

1. Define the design structure and initial state, and
2. Define an evaluation method.

2.2.3 Genetic Algorithms

Genetic Algorithms (“GAs”) are the most popular form of Evolutionary Computation. GAs emphasise the genotypic transformation of individual problem solutions. A typical GA represents a solution to a problem in terms of its genotypic features i.e. the basic features, or elements, that make up a solution. These features are represented using symbolic strings (referred to as “chromosomes”), the most common representation for a GA is a binary (i.e. base 2) string in which each number indicates the presence or absence of a specific element. A set of randomly generated strings is typically used as the initial population for a GA. The chromosomes’ genotypes are then evolved through the application of fitness biased *selection* operators, and recombination and mutation *reproduction* operators.

The steps involved in a typical GA’s evolution can be described more formally using Lewontin’s mappings of biological genetics, as previously described in Figure 2.1:

1. An initial population (g_1) of random binary chromosomes (i.e. genotypes) is created using a random number generator.
2. The genotypes, in the genotypic state space (G), are mapped into the phenotypic state space (P), using a problem specific evaluation function (f_1 *epigenesis*).
3. A subset of the population known as the “gene pool” is selected on the basis of their phenotypes (f_2 *selection*).

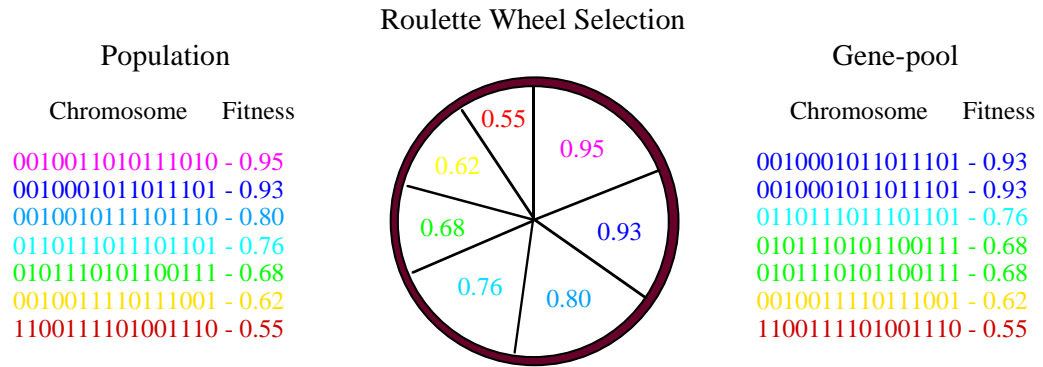


Figure 2.3: An illustration of how “Roulette wheel selection” operates.

4. The chosen phenotypes’ associated genotypes (f_3 *genotypic survival*) in the gene pool are then mated (f_4 *mutation*) to produce a new population of hopefully better solutions (g_1t).

The simulated evolutionary process, involving epigenesis, selection, genotypic survival and mutation, is repeated until an acceptable solution, or set of solutions, to the problem is discovered.

In order to further inform the above overview of GAs, the remainder of this subsection gives some typical selection and reproduction operators used in what is generally referred to as a “standard GA” [Goldberg, 1989], [Davis, 1991], [Mitchell, 1996]. This subsection closes with a description of the tasks involved in applying a GA.

Selection

A typical GA selection operator is “Roulette wheel selection.” This is based on the metaphor of a (fitness biased) roulette wheel in which each chromosome in the population is represented as a slot on a roulette wheel - the area of each slot is proportional to the corresponding chromosome’s fitness rating (see Figure 2.3). Hence, the fitter a chromosome is, the larger its slot will be on the roulette wheel, and therefore the more chance it has of being selected. In order to select a chromosome for reproduction, the wheel is spun and the slot in which the ball lands indicates the chromosome to be selected. A review of this and other forms of selection can be found in [Goldberg, 1989], [Davis, 1991] and [Goldberg and Deb, 1991].

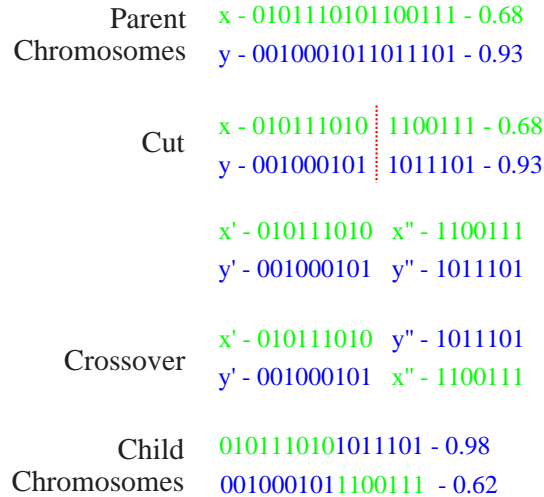


Figure 2.4: An illustration of how “Single point crossover” operates.

Reproduction

Each pair of selected chromosomes reproduce to produce new offspring. As previously noted reproduction involves the use of recombination and mutation operators. “Single point crossover” is a typical recombination operator (see Figure 2.4); two parents, x and y , are split at a random position (i.e. locus) along the chromosome forming four sections; x' , x'' , y' and y'' , the two opposing sections are then crossed and rejoined to form two new chromosomes; $x'y''$ and $y'x''$. A variety of crossover operators are available, see [Goldberg, 1989], [Davis, 1991] or [Pawlowsky, 1995] for detailed reviews.

The second type of operator associated with reproduction is mutation. Within GAs mutation (generally) has a low probability of affecting each symbol in a solution’s string (i.e. each “allele” in a chromosome). When a mutation does occur at a specific chromosome position (i.e. “locus”), it will typically set the allele at that locus to a random symbol in the applied coding alphabet.

Generational and Steady State GAs

GAs typically use “generational” reproduction which evolves by means of replacing the entire population’s chromosomes in one generation with the evolved chromosomes from the subsequent generation. GAs with generational reproduction are used in the examples described in this thesis. “Steady state” reproduction is also used in GAs. Steady state GAs select individual chromosomes for reproduction and (generally) replace the worst individuals with the produced offspring, rather than producing a

complete new set (“generation”) of solutions to replace the previous generation. In several application areas steady state reproduction has been found to be more effective than generational reproduction (see [Whitley, 1988], [Syswerda, 1989], or [Davis, 1991] for further details).

Although the examples given in this thesis refer to generational GAs the visualizations described are equally applicable to steady state GAs. For examples in which the visualizations are indexed by generation number the steady state equivalent would be to index the visualizations by each reproduction replacement event or by a user-specified number of replacement events.

Application

In order to define a GA the user must complete the following three tasks:

1. Define a way of coding (i.e. representing) a state in the problem domain as a string of symbols, referred to as the “genotype” or “chromosome”.
2. Define an evaluation function capable of rating problem states (i.e. chromosomes) in terms of their problem specific behaviour (in phenotypic state space) and returning an appropriate “fitness” score.
3. Define a set of selection and reproduction operators suitable for the problem representation used.

2.3 Summary

The key feature that characterize EP, ESs and GAs is the level in the evolution hierarchy that the algorithms model, that is the *species* in EP, the *individual* in ESs, or the *chromosome* in GAs [Fogel, 1997]. The adaptive species approach of EPs does not use recombination as species of living organisms do not interchange genetic information. Selection with EP is deterministic rather than probabilistic. For EPs the better half of the combined set of parents and children are used as the next population. ESs model evolution at the level of the individual as a process of adaptive behaviour. Solutions are represented as phenotypes which are transformed by a similar set of transformations as used in GAs. GAs model evolution at the level of the chromosome, as adaptive genetics, representing

problem solutions as genotypic chromosomes and applying transformations similar to those observed in the chromosomes of living organisms, such as crossover, inversion and point mutation.

EPs operate through the successive application of mutation and deterministic selection. ESs and GAs, however, both use probabilistic selection biased by the individual's fitness values along with recombination and mutation, the order in which these algorithm components are applied is a distinguishing characteristic between ESs and GAs. GAs are commonly applied in the sequence: selection, recombination, mutation, where as the components of an ES are applied in the sequence: recombination, mutation, selection [Schwefel and Kursawe, 1998].

These different modeling levels have a significant impact on the philosophy that each form of EA adopts to problem solving [Schwefel and Kursawe, 1998]. GA's bottom-up approach, views crossover as a method for combining "good genes" from existing solutions to produce better solutions. The recombination of good genes to produce better solutions is referred to as the "building block hypothesis." This hypothesis has drawn a considerable amount of controversy within the EC community as some consider it improper to attribute any form of quality to a chromosome's components when it is only possible to evaluate the quality of the chromosome as a whole whose worth is realized through the purposeful interactions of its components [Fogel, 1998b]. Although there is some empirical evidence that indicates some types of evolutionary search algorithm are better than others for certain types of problem (see [Fogel, 1993], [Fogel, 1994], [Rizki et al., 1993]), there is also sound theoretical evidence that there is no single EA which is superior in all problem domains (see [Wolpert and Macready, 1997] and [Fogel and Ghozeil, 1997]).

This project explores GA visualization as a specific case study of applying Software Visualization ("SV") technology to EC. The aim here is to explore how SV can support peoples' understanding of GAs, and through the recognition of the differences between GAs and other forms of EC, propose a series of generic recommendations for EC visualization. The following set of key GA characteristics were used as a starting point for exploring how SV may be applied to support the GA user's understanding of their algorithm's search behaviour:

1. *The operation of the GA's component parts*; the operation of the selection operator, the operation of the crossover operator, or the operation of the mutation operator.

2. *The quality of the solutions found by the GA*; all of the chromosomes' fitness ratings, a user defined subset of the chromosomes' fitness ratings, or the best, average and/or worst fitness rating in each population.
3. *The chromosomes' genotypes*; all the chromosomes' genotypic representations in a specific generation, a user defined subset of the chromosomes in the current generation, or the best chromosome in the current generation.
4. *The chromosomes' phenotypes*; all the chromosomes' phenotypic traits in a specific generation, a user defined subset from the current generation, or the best chromosome's phenotype in the current generation.
5. *The GA's sampling of the search space*; the diversity of the chromosomes' genotypes in a specific generation, or the GA's coverage of the search space during the GA's run.
6. *Navigating the GA's execution*; moving backwards and forwards through each generation of the GA's run.
7. *Editing the GA*; editing the algorithm's parameters, editing the algorithm's components, or editing the population's chromosomes.

Chapter 3

GA User Study

By gaining an understanding of the current working practices of GA users, visualizations can be constructed to support or modify the users' working practices. The study described in this chapter provides a set of visualization requirements for GA users. The study design is described in Section 3.1, the results are presented and discussed in Section 3.2, and the conclusions of the study, identifying a set of GA visualization requirements, are presented in Section 3.3. A copy of the study materials, a summary of the responses, and the respondents' completed questionnaires are given in Appendices A, B, and C, respectively.

3.1 Design

The purpose of this study was to identify the visualization requirements of GA users. In order to do this a questionnaire was designed to discover the current working practices of GA users, the difficulties they encounter, and any opinions or suggestions they may have regarding GA visualization.

Labaw's recommendations for questionnaire design [Labaw, 1980] focus upon testing hypotheses. Specifically the *client hypotheses*: the client's reasons for commissioning the study and the intended use of the results; the *professional hypotheses*: ensuring that the respondent is treated ethically; and the *research hypotheses*: regarding the nature of the problem, the nature of the respondents, the sample design, and the topics covered in the questionnaire. The purpose for Labaw's strict hypothesis approach is to ensure that the questionnaire is designed with an explicit goal in mind and that any

assumptions are explicitly expressed prior to the study being applied. When the purpose of the study is to test and verify something, then recording all of the questionnaire designer's assumptions is a necessary stage in producing an accurate test method.

However, when the purpose of the study is to explore and gather new knowledge rather than test existing knowledge, it is inappropriate to form any pre-study assumptions regarding the respondents' replies - by the very nature of exploration the study instrument should be open to unexpected responses. So although it is still valid to note the assumptions made regarding the client and professional ethics, pre-study research hypotheses regarding the nature of the problem are inappropriate in this case. Instead some tentative proposals regarding the issues involved are introduced to elicit the respondents opinions.

The following subsection (3.1.1) describes the issues tackled in the study as well as the intended use of the results and the professional ethics involved. Having identified the study issues, an electronic questionnaire was chosen as an appropriate study mechanism. Subsection 3.1.2 describes the questionnaire's delivery, and subsection 3.1.3 describes the questionnaire's structure and content. Hard-copies of the e-mail message introducing the questionnaire and the questionnaire itself are provided separately in Appendix A.

3.1.1 Issues

This subsection examines four aspects of the study's design; the purpose of the study, the ethics involved, the nature of the respondents, and the topics addressed. With regard to the professional ethics extended to the respondents two issues were considered important; the time taken to complete the questionnaire and the confidentiality of the results. It was decided that an upper bound on the time taken to complete the questionnaire should be set at 30 minutes. Although it was thought that the respondents may be reluctant or embarrassed to identify any difficulties they have using GAs, the respondents may need to be contacted in the future, in order to clarify their responses or evaluate the resulting system. The respondents were asked to supply their name and email address. Every assurance was given that their responses would be held in confidence and that their identity would not be included in any associated publications.

A number of assumptions regarding the nature of GA users had to be made in order to decide

which study mechanism and delivery method to apply. From previous work on GAs and from browsing recent conference proceedings it was known that GA users work in both industry and academia, and that although the popularity of GA research within the UK is growing, GA users are not located exclusively within the UK but are located throughout the world. It was also evident that GA users have easy access to computers and the internet (including access to email, email-based discussion groups, and the world wide web).

The goals of the study were clear; to identify the working practices of GA users and their opinions with regard to GA visualization and human-GA interaction. These three goals were used to form the basic structure of the study. Two additional topics were added, one to identify the respondent's background, i.e. experience, motivation and environment, and one to request permission to contact the respondent in the future with regard to this project. The sections of the study were designed to address these five topics, namely; "Background Information," "Your Approach to GAs," "What Characteristics to Visualize," "Interaction Opportunities," and "Future Contact."

3.1.2 Delivery

Having assumed that the majority of GA users are not based in the UK, the use of face-to-face interviews or telephone interviews was deemed impractical. However, given that the same people were assumed to have easy access to computing equipment and the internet, an online electronic questionnaire was considered to be an appropriate query mechanism. When compared with traditional postal questionnaires this had the additional benefits of low-cost and easy-distribution.

An HTML¹ version of the questionnaire was constructed and placed on the internet. Links to the HTML version of the questionnaire were made from the author's home page and progress report page, as well as from the EvolutioNary COmputation REpository network ("ENCORE"). The editor of ENCORE included the following link and encouraging message on the ENCORE home page:

"GA/Viz: Scientific Visualization of Genetic Algorithms a questionnaire compiled by Trevor Collins. [I really hope something comes out of this project; so take some time off and insert "all the ideas for graphical gimmicks you always wanted to put into your apps but never had the time (or knowledge) to" into here, folks! -Ed.]"

¹HTML stands for the HyperText Mark-up Language, the language used to create world wide web pages.

A message was posted to three Usenet bulletin boards that frequently carry messages regarding EC (i.e. comp.ai, comp.ai.alife and comp.ai.genetic, see Appendix A). The message was in the style of a covering letter that contained the internet address for the questionnaire and a plain text version of the questionnaire. Although the contents of both hypertext and plain text versions of the questionnaire were identical, it was thought that some people may find it easier to simply reply to the posted message and include their response, rather than go to the internet address and fill out the online questionnaire.

Although the postings on the related bulletin boards and internet repository provided an opportunity for interested parties reading the messages to complete the questionnaire, a more direct approach was considered necessary in order to encourage a greater response. An email message similar to the Usenet posting was used for this purpose. This message was sent to several representatives of known GA research groups, namely the Illinois Genetic Algorithm Laboratory (IlligAL), the Genetic Algorithms Research Group at the University of Michigan (GARG), the Genetic Algorithm Research and Applications Group at Michigan State University (GARAGe), and the Genetic Algorithms Group at George Mason University. Each representative was asked to circulate the questionnaire to any interested members of their group as well as completing it themselves. Direct emailing was carried out to a further ten published researchers whose publications indicated an interest in the application of GAs or the visualization of GAs.

3.1.3 Structure and Content

As the purpose of this study was to explore the working practices and opinions of GA users, open-ended free response questions were used throughout. For the hypertext version of the questionnaire some details about the purpose of each question were included in a separate HTML page, available to the respondent via hypertext links in the introductory paragraphs of the questionnaire. In addition to the five study sections identified in the previous subsection, a penultimate question was introduced before the “Future Contact” section asking the respondent to add any further comments they may have on how to make GAs easier to use. The remainder of this subsection describes the design of each question.

Background Information

Prior to asking the respondent to comment on their approach to applying GAs, or GA visualization, a few introductory questions were asked in order to establish the respondent's profile (i.e. demographics). Each respondent was asked the following four questions:

1. How long have you been using GAs?
2. During this time what have you used GAs for?
3. Why did you use GAs for these tasks?
4. What environment(s) do you use when working with GAs? Please specify each computing environment separately i.e. the computer system, programming language and/or application tool.

This information is necessary in order to explain why people are using GAs as a preferred means for solving their respective problems and to identify if people's working practice is dependent on the amount of experience they have, the area of application they work in, or the environment being used.

Your Approach to GAs

As noted in the introduction (Chapter 1), in order to apply a GA the user must;

- Define a way of representing a state in the problem domain as a string of numbers referred to as the chromosome's "genotype."
- Define an evaluation function capable of rating the genotypes in terms of their problem specific behaviour (i.e. phenotypic traits).
- Define a set of selection and reproduction operators suitable for the problem representation used.

Question 5 asked the respondents to identify any difficulties they had with these three tasks and with selecting suitable algorithm parameters. Finally, in order to establish if GA users complete any other tasks whilst applying GAs, the respondents were also asked to specify any additional set-up steps they deploy prior to running their algorithms and any difficulties they associate with those steps:

5. What do you find difficult, if anything, about the following set-up steps involved in creating a GA:
 1. Defining the mapping between the problem domain and the string representation used by the GA?
 2. Producing an effective evaluation function?
 3. Choosing the GA's components, e.g. the initial population creation method, what reproduction gene-pool selection criterion to adopt, which genetic operators to apply, etc.?
 4. Selecting suitable parameters for the GA, e.g. the population size, the mutation rate (if appropriate), etc.?
 5. Are there any other set-up steps that you use before running the GA? If so please note them and any associated difficulties you encounter below.

After applying a GA the user can easily identify the best solution as being the fittest chromosome discovered during the algorithm's search. However, this fails to take into account the quality of the search pattern and the quality of the other solutions considered. The steps involved in establishing the quality of the search and the solution(s) found are not explicitly described in any of the introductory literature. In order to find out just what people do in terms of quality assurance, Question 6 asked the respondents to describe the approach they took:

6. Having applied a GA to a particular problem what approach do you take, in order to:
 1. Assess the quality of any solution(s) found?
 2. Examine how representative the output of the GA is in terms of all the possible points within the problem-space?

What Characteristics to Visualize

Having enquired about the difficulties people have applying GAs and the steps they perform, the questionnaire moved on to examine some aspects of GA visualization. Although SV is explicitly intended to support the user, the additional time and effort involved in constructing and viewing

such visualizations may well detract from their usefulness. In order to identify peoples opinions toward SV, Question 7 asked the respondent to comment on both the advantages and disadvantages of representing different aspects of their algorithm's data:

7. If the following typical output characteristics were to be represented what advantages or disadvantages, if any, could you foresee?

1. All of the individual chromosomes within each population.

Advantages:

Disadvantages:

2. A user defined selection of representative chromosomes.

Advantages:

Disadvantages:

3. The rate of change in the population's fitness values, i.e. the gradient values of a fitness versus generation graph.

Advantages:

Disadvantages:

Question 8 then asked a similar set of questions about representing additional information to the typical output, such as the actions of the genetic operators or summary statistics regarding the population's diversity:

8. As well as directly illustrating the output of the GA, visualization could be used to represent additional information either derived from the output dataset or recorded separately. If visualization were used to represent the following characteristics what advantages or disadvantages, if any, could you foresee?

1. The chromosomes in the reproduction gene-pool.

Advantages:

Disadvantages:

2. The occurrence of mutation in chromosomes where a mutation operator has been applied.

Advantages:

Disadvantages:

3. The internal actions of the genetic operators being applied to the chromosomes, e.g. the splitting and crossover between two chromosomes by a single point crossover operator.

Advantages:

Disadvantages:

4. A “similarity” rating for each chromosome based on how little they differed to the fittest chromosome, e.g. a ten bit binary chromosome that differed from the fittest chromosome in three of its bit positions (“loci”) may have a similarity rating of 0.7.

Advantages:

Disadvantages:

The suggestions for GA visualization presented in Questions 7 and 8 were derived from some of the existing GA visualization techniques and systems described in Section 4.2.1. Question 9 asked the respondent to specify any other characteristics that they would like to see visualized:

9. Please specify any other direct or indirect characteristics that you would be interested in seeing visualized.

Interaction Opportunities

As well as presenting information about a GA’s execution, SV is concerned with ways in which user interaction can be used to aid understanding. In the case of EC this offers the user an opportunity to interact with the search data and the search algorithm. However, exactly how helpful the different levels of interaction will be is difficult to predict. SV systems typically support the use of interactive control mechanisms and the editing of an algorithm’s parameters. Question 10 asked the respondents to comment on how helpful or destructive they would find the use of an interactive control panel, an

editor for altering their algorithm's parameters, an editor for editing the chromosomes in the current population, and an editor for editing the chromosomes in the reproductive gene pool.

10. How helpful, or destructive, would you find each of the following interaction opportunities for your use of GAs?
 1. Execution control through the use of a control panel to run, pause, step forward, step backward, save a snapshot, and/or stop execution.
 2. Editing the algorithm's parameters during execution.
 3. Editing the population's chromosomes between two generations.
 4. Editing the reproduction gene-pool's chromosomes within a generation.

Although Question 10 raised the issue of interaction, the level of interaction suitable for the user is expected to depend upon the task they are carrying out. Question 11 asked the respondent to specify any other forms of interaction which they would consider helpful.

11. Please specify any other forms of additional interaction that you would consider beneficial.

Any Other Comments

In order to provide the respondent with a final opportunity to add any other suggestions they may have for making GAs easier to use, Question 12 asked the respondent for any other comments:

12. Do you have any other suggestions on how GAs could be made easier to use? Or any other comments at all about GAs? Please note them below.

Future Contact

Finally, in case any of the respondents' replies were ambiguous or would require further discussion, and, in order to establish an opportunity for future evaluation of the resulting visualizations, the respondents were asked if they had any objection to being contacted in the future:

13. Finally, would you have any objection to being contacted in the future with reference to this project and the evaluation of the resulting GA visualization system?

Table 3.1: Question 1. A summary of the amount of time the respondents had been using GAs.

DURATION (YEARS)	USERS	INDIVIDUAL RESPONDENTS
$0 \leq 1$	3	<i>R1 R2 A1</i>
$1 \leq 2$	7	<i>T1 R3 R4 R5 R6 A2 A3</i>
$2 \leq 3$	4	<i>T2 R7 A4 A5</i>
$3 \leq 4$	3	<i>T3 A6 A7</i>
$4 \leq 5$	1	<i>R8</i>
$5 \leq 6$	0	
$6 \leq 7$	1	<i>A8</i>

- Yes. I would object to being contacted in the future.
- No. I would not object to being contacted in the future.

3.2 Results and Discussion

This section discusses the responses given in the study. A synopsis of the responses is given in Appendix B and an anonymised copy of each respondent's completed questionnaire is included in Appendix C. Nineteen completed questionnaires were received. This section presents a summary of the comments made and discusses the opinions expressed.

Although the sample size was too small to make any statistical predictions regarding the GA community as a whole, that does not detract from the validity of the opinions expressed. These results are used to identify a set of visualization requirements which in turn are used to guide (but not constrain) the design of the GA visualization support provided in this thesis.

3.2.1 Background Information

The respondents' backgrounds were recorded by the length of time they had worked with GAs, their motivation and use of GAs, and their computing environment (see Appendix B). The duration of use varied from 2 months to 7 years, see Table 3.1 for details. The distribution of the respondents'

duration of usage was not sufficient to form experience based groupings (such as naive, novice and expert). Groups made solely on the duration of experience were considered inappropriate for these users as their concentration of work done with GAs varied considerably. Some used GAs sporadically for university projects, some used GAs for specific problems, and others worked with GAs exclusively as the focus of their work.

The respondents' area of application varied widely. Examples included research on different problem representations, the role of mutation, neural network, silicon chip and electronic circuit design, routing problems, and protein sequencing problems. There were too few common domains to derive any application oriented categories, although (when relevant), the respondents' motivation was used to form groupings. These respondents were motivated either by their interest in GAs, or by a need to solve a problem for which GAs were an appropriate problem solving method. Those that were interested in GAs could also be further divided into those working on the problem-independent theoretical aspects of GAs and those working on the application of GAs to specific problem domains.

Those respondents interested primarily in the theory of GAs are referred to as the "GA theory group" (3 people). Those concerned in the application of GAs but as a direct result of their interest in GA research are referred to as the "GA research group" (8 people). Thirdly, the respondents concerned primarily with solving a problem, for which GAs offered an effective approach are referred to as the "GA applications group" (8 people). In the results reported here the respondents are referred to by a motivation group letter (*T*, *R*, or *A*) and a respondent number ($0, 1, 2, \dots N$). These motivational factors were reflected in the respondents' responses regarding their working practices and visualization requirements.

Thirdly, the respondents' computing environments were recorded. The majority of these respondents were familiar with UNIX based workstations and wrote their algorithms in C or C++ (see Table 3.2). All of the respondents indicated that they were highly computer literate, often experienced with more than one machine and more than one language.

3.2.2 Your Approach to GAs

The respondents' motivation did appear to have a profound effect on their approach to using GAs. In this section the differences between the opinions expressed by some of the theory group respondents

Table 3.2: Question 4. A tally of the environments used by the respondents. Showing the distribution of respondents using each machine, language and toolkit identified in the responses.

ENVIRONMENT	USERS	INDIVIDUAL RESPONDENTS
<i>Machine</i>		
UNIX	16	<i>T3 T1 T2 R5 R7 R2 R6 R1 R4 R3 A2 A4 A5 A8 A3 A6</i>
DOS	6	<i>T3 T2 R4 A2 A1 A7</i>
PC	2	<i>R4 R3</i>
Macintosh	1	<i>T1</i>
VMS	1	<i>A3</i>
Amiga	1	<i>A1</i>
Various	1	<i>R8</i>
<i>Language</i>		
C	12	<i>T3 T1 R5 R7 R1 R4 A2 A4 A1 A5 A8 A3</i>
C++	7	<i>T1 T2 R2 R6 R1 A2 A4</i>
Matlab	2	<i>R3 A6</i>
Pascal	1	<i>A7</i>
Fortran	1	<i>A1</i>
Pop11	1	<i>A5</i>
Smalltalk	1	<i>T2</i>
Various	1	<i>R8</i>
<i>Toolkit</i>		
GAMETER	1	<i>R4</i>
GA MATLAB TOOLBOX	1	<i>R3</i>

Table 3.3: Question 5.1. A tally of comments made by the respondents with regard to the difficulties they encountered whilst defining the mapping between the problem domain and the string representation used by the GA. The respondents stated that was either a difficult task for them, not difficult for their particular problem, or important to the outcome of the GA. Note, *R4* considered this to be both difficult and important, hence the double entry.

REPRESENTATION	USERS	INDIVIDUAL RESPONDENTS
<i>Difficult</i>	8	<i>T3 R2 R8 R5 A6 A7 R4 A3</i>
<i>Not Difficult</i>	7	<i>R6 R1 R3 R7 A2 A1 A4</i>
<i>Important</i>	3	<i>T1 R4 A8</i>

Table 3.4: Question 5.2. A tally of the comments made regarding the evaluation function's definition. In addition to either being difficult or not difficult, some respondents noted that this was a difficult task only when there are conflicting criteria for evaluating the GA's solutions to the problem, or that they were not interested in the development of the evaluation functions.

EVALUATION	USERS	INDIVIDUAL RESPONDENTS
<i>Difficult</i>	10	<i>R5 R7 R3 R8 R6 A1 A4 A8 A7 A6</i>
<i>Difficult - Conflicting Criteria</i>	2	<i>T2 R2</i>
<i>Not Difficult</i>	4	<i>R1 R4 A2 A3</i>
<i>Not Interesting</i>	2	<i>T1 T3</i>

and those of the research and applications groups were quite distinct, highlighting the different goals of each group of users (see Appendix B for the contrasting comments).

The members of the theory group were interested primarily in their understanding of GAs; their application of GAs was to further their understanding through experimental testing. Although asked specifically to identify any difficulties they experienced, the theory group respondents did not explicitly discuss their problems. These users more commonly identified how important or how interesting they found each step (see Tables 3.3, 3.4, 3.5 and 3.6). There was very little consensus of opinion between these three respondents other than the importance of finding an effective representation scheme.

Table 3.5: Question 5.3. A tally of the comments made regarding the selection of the GA's components. As well as identifying this as either a difficult or not difficult task, some of the respondents considered this to follow directly from the representation they used and therefore not difficult, or they were unaware of the best components and used either default components that had worked in the past, followed existing guidelines published in the literature, or made modifications through unguided trial and error.

ALGO. COMPONENTS	USERS	INDIVIDUAL RESPONDENTS
<i>Not Difficult</i>	2	<i>A2 A6</i>
<i>Follows From Problem Rep.</i>	2	<i>R4 R8</i>
<i>Difficult</i>	4	<i>T1 R2 R5 R1</i>
<i>Use Defaults</i>	4	<i>T3 R3 R6 A1</i>
<i>Follow Published Guidelines</i>	3	<i>R7 A7 A3</i>
<i>Use Trial and Error</i>	1	<i>A8</i>

The research group were working on solving real-world problems but they too were motivated by their interest in GAs. The research group respondents either felt that their problems were easy to represent, or extremely difficult to represent and that this was entirely dependent on the problem domain (Table 3.3). A similar problem-dependent view was taken with the definition of the evaluation function, either the evaluation function was obvious given the problem objective and representation used, or difficult particularly for problems involving multiple performance measures (see Table 3.4). Views on the selection of algorithm components and parameter settings were closely tied, the same three research group respondents experienced difficulties with both. Several research group respondents linked their choice of algorithm components to the problem representation used. Some respondents used default algorithm components and parameters, others used interactive tools to change their operators and/or parameters during their algorithm's execution, yet no clear principles for GA design were raised.

The members of the applications group were focused specifically on the problem being solved, they had little or no vested interest in GA research or GA theory. The applications group respondents used GAs as an "off-the-shelf" tool, they made few changes to their algorithms' components, and any changes they made to their parameter settings were typically made through unguided trial and

Table 3.6: Question 5.4. A tally of the comments made regarding the selection of the GA's parameters. The comments made here are similar to those made in Table 3.5.

ALGO. PARAMETERS	USERS	INDIVIDUAL RESPONDENTS
<i>Not Difficult</i>	3	<i>R8 A3 A1</i>
<i>Difficult</i>	7	<i>T2 R2 R1 R5 A2 A6 A5</i>
<i>Use Defaults</i>	4	<i>T3 R3 R4 R6</i>
<i>Use Trial and Error</i>	3	<i>A8 A7 A4</i>
<i>Follow Published Guidelines</i>	1	<i>R7</i>
<i>No Opinion</i>	1	<i>T1</i>

Table 3.7: Question 6.1. A tally of the steps carried out by the respondents to verify the quality of the solutions found by the GA. Specifically, respondents either did nothing more than check the fitness ratings of their solutions, verified their results by repeatedly running the algorithm, or compared the results of their algorithm to those of other alternative approaches. Note, respondent *A8* carried out both comparisons against the results of other algorithms and repeated runs.

STEPS TAKEN	USERS	INDIVIDUAL RESPONDENTS
<i>Check Fitness</i>	9	<i>R2 R8 R6 R3 R5 A6 A5 A1 A3</i>
<i>Repeated Runs</i>	6	<i>T2 T1 R7 A2 A7 A8</i>
<i>Compare Against Other Algorithms</i>	3	<i>T3 R4 A8</i>

error (see Tables 3.5 and 3.6). The fact that these users were applying GAs because few or no other techniques are effective at solving their problem, indicates the complexity of the problem domain. When designing effective methods for evaluating their chromosomes the biggest difficulty that these users experience is identifying what constitutes a “good” chromosome (see Appendix B). Representing solutions to these problems in abstract strings of symbols is much less of a problem because although these users may have problems evaluating their solutions, their expertise in the problem domain gives them a clear knowledge of what may be involved in solving the problem, and therefore, what features should be included within the problem representation.

When exploring people’s attitudes toward quality assessment it was surprising that only six of

Table 3.8: Question 6.2. A tally of the steps carried put by the respondents to verify the quality of the GA's search of the problem space. The respondents either did nothing to explore the quality of their GA's search, or they examined the proposed solutions, compared the solutions given by alternative approaches, or in one case, explored the variation in fitness across the search space.

STEPS TAKEN	USERS	INDIVIDUAL RESPONDENTS
<i>Do Nothing</i>	6	<i>R6 R4 A1 A5 A2 A7</i>
<i>Examine Proposed Solutions</i>	4	<i>R3 R1 R8 R5</i>
<i>Compare Against Other Algorithms</i>	1	<i>T1</i>
<i>Examine Fitness Surface</i>	1	<i>A8</i>

the nineteen respondents attempted to explore the diversity of the search space sampled by their GA given that premature convergence is widely recognized as a significant problem for simulated evolution (see Table 3.8). Although the user's intuition and knowledge of the problem domain may well serve as a sanity check for the results of the GA, ignoring the GA's search path leaves the user without an understanding of the GA's actions and without access to any of the potential sources of error.

The respondents' use of default algorithm components and parameter settings, and their use of trial and error for improving their algorithms' performance (as highlighted in Question 5), implies that the effects of different algorithm components and parameter settings are not fully appreciated by everyone within the GA community. Moreover, the fact that less than half of the respondents took any additional steps to verify their results (other than to examine the fitness of the evolved solutions) and less than a third made any attempt to examine their algorithm's coverage of the search space, indicates that GAs are being applied as a "black-box" problem solving method, taking inputs in the form of the problem representation and evaluation function, and outputting a set of possible problem solutions. Owing to the fact that GAs are so robust the black-box approach often produces effective results, however, the user does not know how the solutions are produced and is therefore unable to make improvements to the design or track down any possible causes of error.

Table 3.9: Question 7.1. The reported Advantages and Disadvantages of visualizing the individual chromosomes in the population. This was generally considered useful for seeing different aspects of the population, although some respondents considered this be of no advantage. The disadvantages noted here relate to the scalability of the visualization, in that it may present too much information and slow down the GA, or the information presented may confuse the user or may be difficult to represent.

VIZ CHROMOSOMES	USERS	INDIVIDUAL RESPONDENTS
<i>Can already do it</i>	1	<i>T2</i>
ADVANTAGES		
<i>None</i>	3	<i>T1 A3 A8</i>
<i>A Lot of Information</i>	4	<i>R1 R3 A1 A5</i>
<i>See Whats Happening</i>	4	<i>T3 A7 A6 A2</i>
<i>See individual Differences</i>	3	<i>R6 R7 A4</i>
<i>See Population's Convergence</i>	2	<i>R6 R4</i>
<i>See if Stuck in Local Optima</i>	2	<i>R2 R4</i>
DISADVANTAGES		
<i>Scalability - Too Much Information</i>	14	<i>T1 T3 R1 R3 R6 R4 R8 R5 R2 A8 A7 A6 A2 A5</i>
<i>Might Confuse People</i>	4	<i>R4 R8 A7 A1</i>
<i>Scalability - Too Slow</i>	1	<i>R7</i>
<i>Representation Problems</i>	1	<i>R2</i>

3.2.3 What Characteristics to Visualize

The respondents' attitude toward visualization was generally very positive with a preference for showing high-level (i.e. macro-level) abstractions of the GA's behaviour, rather than its low-level (i.e. micro-level) internal operations. Although fitness versus generation number graphs are widely used to illustrate the progress of a GA, these respondents also wanted to see the algorithm's behaviour and the diversity of the solutions found.

The respondents were generally in agreement that visualizing every chromosome in each population would be helpful for seeing what was going on within the GA, but they also felt that this would produce too much information to be of use (see Table 3.9). Enabling the user to select a subset of solutions

Table 3.10: Question 7.2. The Advantages and Disadvantages of visualizing a user defined selection of the chromosomes in the population. Again like Table 3.9, the advantages of viewing selected chromosomes relate to seeing what is happening in the population, this has the added advantage of being flexible to the user's requirements but also the disadvantage of perhaps under representing the chromosomes in the population and thereby confusing the user.

VIZ SELECT CHROMOSOMES	USERS	INDIVIDUAL RESPONDENTS
<i>Can already do it</i>	1	<i>T2</i>
ADVANTAGES		
<i>Reduces the Scaling Problems of 7.1</i>	6	<i>R6 R1 R4 A6 A1 A8</i>
<i>See What's Happening</i>	3	<i>R3 A2 A7</i>
<i>See Individual Differences</i>	1	<i>A4</i>
<i>See Emergence of Niches</i>	1	<i>A3</i>
<i>Establish Viewing Conventions</i>	1	<i>R5</i>
<i>Flexible to User Requirements</i>	1	<i>T3</i>
<i>Helpful</i>	1	<i>A5</i>
DISADVANTAGES		
<i>None</i>	1	<i>A8</i>
<i>Might Not be Representative</i>	8	<i>T1 R6 R1 R4 R2 A1 A4 A5</i>
<i>Might Confuse People</i>	3	<i>R8 A1 A4</i>

from each population would reduce the information overload but this itself introduced problems regarding the user's ability to select a representative sample and not miss important chromosomes in the population (see Table 3.10).

The need for viewing information (in this case the chromosomes) at an effective level of abstraction is a commonly tackled problem in SV. The solution proposed by [Eisenstadt et al., 1990] is to provide coupled views that give both coarse grained and fine grained perspectives. For example, a "goal tree" metaphor is used in the Transparent Prolog Machine ("TPM") [Eisenstadt and Brayshaw, 1987], to couple TPM's coarse grained and fine grained views of a Prolog program's goals (see Section 4.2.2).

Visualizing the rate of change in the population's fitness was considered useful by most of the

Table 3.11: Question 7.3. The Advantages and Disadvantages of visualizing the rate of change in the populations' fitness ratings. The additional comments here relate to the user's interest in seeing more than just the chromosomes' fitness ratings, they also need to see how fitness relates to the local structure of the chromosomes. Concerns were also noted with regard to the effect visualization would have on the speed of the GA.

VIZ FITNESS CHANGES	USERS	INDIVIDUAL RESPONDENTS
<i>Not Sure</i>	1	<i>A1</i>
ADVANTAGES		
<i>Not Enough Information</i>	2	<i>T3 T1</i>
<i>Not Useful - Fitness Too Noisy</i>	1	<i>R8</i>
<i>Not Necessary</i>	1	<i>R3</i>
<i>Shows What's Happening</i>	5	<i>T2 A8 A6 A7 A4</i>
<i>Shows Convergence</i>	5	<i>R6 R7 R1 R4 A5</i>
<i>Shows Population's Stability</i>	1	<i>R1</i>
<i>Shows if Population Stagnating</i>	1	<i>R2</i>
<i>Essential</i>	1	<i>R5</i>
DISADVANTAGES		
<i>Can't Think of Any</i>	1	<i>R2</i>
<i>Not Enough Information</i>	2	<i>R4 A8</i>
<i>Not Useful - Fitness Too Noisy</i>	1	<i>R6</i>
<i>Scalability - Too Slow</i>	1	<i>T2</i>
<i>Representation Problems</i>	1	<i>T3</i>
<i>Only Useful in Support of Fitness v Time Graph</i>	1	<i>R1</i>

Table 3.12: Question 8.1. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool. Comments were with regard to validating the correct operation of the GA, also for problems in which the genotype is meaningless such views were considered to be not informative, not meaningful or not of interest to the user.

VIZ GENE-POOL CHROMOSOMES	USERS	INDIVIDUAL RESPONDENTS
Advantages		
<i>See What's Happening</i>	3	<i>R6 R7 R4</i>
<i>Validate Algorithm</i>	2	<i>T2 A7</i>
<i>Depends on Problem</i>	2	<i>R2 A8</i>
<i>Shows Convergence</i>	1	<i>R5</i>
<i>Not Informative</i>	1	<i>T3</i>
Disadvantages		
<i>None</i>	1	<i>A8</i>
<i>Not Meaningful</i>	3	<i>R8 R4 A5</i>
<i>Representation Problems</i>	3	<i>R2 R6 R1</i>
<i>Not Interested</i>	1	<i>R3</i>

respondents, although some confusion regarding the interplay between visualizing the fitness ratings and visualizing the rate of change in fitness did arise (see Table 3.11). Four of the respondents (theory 2/3, research 1/8, applications 1/8) stated that they often wanted more information about the local structure of the populations than fitness graphs could give.

The visualization of the chromosomes in the reproductive gene-pool, the occurrence of mutation, and the internal actions of the genetic operators, were generally considered helpful for illustrating the operation of the GA but useful only as a teaching or debugging aid (Tables 3.12, 3.13, and 3.14). The majority of these comments were made by the research group; the theory and application group respondents could see little use for these three visualizations which illustrate at a micro-level the actions performed by the GA. Details at this fine-grained level are of little interest to the user unless they need to explore the actions of each algorithm component, either to illustrate the algorithm's operation to others (i.e. for teaching), or to locate a bug (i.e. for debugging).

Table 3.13: Question 8.2. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the occurrence of mutation in chromosomes. The comments made here were similar to previous comments, one of the respondents also considered this to involve a high computational overhead for a relatively small contribution to their understanding.

VIZ MUTATION	USERS	INDIVIDUAL RESPONDENTS
Advantages		
<i>Unsure</i>	1	<i>R2</i>
<i>See What's Happening</i>	4	<i>R6 R4 A6 A3</i>
<i>Entertaining</i>	1	<i>T3</i>
<i>Validate Algorithm</i>	1	<i>A7</i>
<i>Not Useful - Too Much Information</i>	1	<i>T2</i>
<i>Not Interested</i>	1	<i>R3</i>
<i>None</i>	1	<i>R5</i>
Disadvantages		
<i>None</i>	1	<i>R6</i>
<i>Not Very Informative</i>	1	<i>A5 A8</i>
<i>Need to be Selective</i>	1	<i>T3</i>
<i>A Distraction</i>	1	<i>R5</i>
<i>Not Meaningful</i>	1	<i>R8</i>
<i>A Lot of Overhead</i>	1	<i>R4</i>

The fourth suggested visualization, the illustration of a similarity rating for each chromosome based on the chromosome's Hamming distance² from the fittest, was considered by members of all of the respondent groupings to be a useful view (Table 3.15). The only concern raised was with regard to the effectiveness of such a measure. However, if introduced as part of a set of problem specific similarity measurements this would be one way of enabling the user to explore the local structure of the fitness changes within a population.

The last question in this visualization section (Question 9), asked the respondents to describe any

²The "Hamming distance" is the total number of differing bits in a string e.g. the Hamming distance from 1001010101 to 1001000111 is 2.

Table 3.14: Question 8.3. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing the internal actions of the genetic operators.

VIZ OPERATORS	USERS	INDIVIDUAL RESPONDENTS
Advantages		
<i>None</i>	2	<i>T2 R8</i>
<i>See What's Happening - Operators</i>	4	<i>R6 R4 A6 A8</i>
<i>Useful - Education & Debugging</i>	4	<i>R7 R2 R5 A7</i>
<i>Interesting</i>	1	<i>R1</i>
Disadvantages		
<i>Should be Optional</i>	2	<i>T3 R2</i>
<i>None if Optional</i>	1	<i>R5</i>
<i>Unnecessary</i>	1	<i>R6</i>
<i>A Lot of Overhead</i>	1	<i>R4</i>
<i>Not Interesting</i>	1	<i>R3</i>

other visualizations they considered useful. The responses given, see Appendix B, were surprisingly varied, emphasizing the need for a flexible visualization environment in which the users can construct their own visualizations specific to the algorithm they are using or the problem-domain being explored.

Overall, of the three direct visualizations suggested in Question 7, viewing every chromosome produces too much information, a user defined sub-set may miss out some important information, and fitness related graphs do not give enough detailed information. These comments can be used to focus the goals of GA visualization; clearly more views are needed to illustrate the content of each population in a form that explains the local structure of the fitness changes throughout the population but at a sufficient level of abstraction to avoid too many unnecessary details. From the derived visualizations suggested in Question 8, the 3 micro-level visualizations of the GA's internals were not considered useful other than as an aid for teaching or debugging, but macro-level visualizations of similarity measures, such as the Hamming distance to the fittest chromosome, were considered to give a useful insight into the diversity of the population.

Table 3.15: Question 8.4. A tally of the respondents' comments regarding the advantages and disadvantages of visualizing a "similarity" rating for each chromosome.

VIZ CHROMOSOME SIMILARITY	USERS	INDIVIDUAL RESPONDENTS
<i>I Use Fitness Similarity</i>	1	<i>T2</i>
Advantages		
<i>Important - Very Good</i>	7	<i>T3 R7 R5 R1 R3 A3 A8</i>
<i>Shows Convergence</i>	3	<i>R6 R2 A7</i>
<i>Shows Diversity</i>	1	<i>A6</i>
<i>Good for Educational Purposes</i>	1	<i>A1</i>
Disadvantages		
<i>None</i>	2	<i>R6 A8</i>
<i>Needs More Sophistication</i>	2	<i>T3 R8</i>
<i>Problem Dependent</i>	1	<i>R4</i>
<i>Might be Confusing</i>	1	<i>A7</i>
<i>Slows GA Down</i>	1	<i>A1</i>

3.2.4 Interaction Opportunities

Within a visualization environment interaction can be used in many ways, for example, to navigate the GA's execution, to alter the algorithm design (such as the parameter settings and algorithm components), or to alter the chromosomes in the population. These three examples illustrate three different levels of intervention.

The use of a bi-directional control panel and an online parameter editor were both considered very useful (see Tables 3.16 and 3.17). No disadvantages were reported regarding the use of a bi-directional control panel and there was only one disadvantage regarding the use of the parameter editor. This disadvantage suggested that rather than editing the parameters manually an adaptive parameter scheme should be included within the algorithm design. This is certainly a valid point: adaption schemes have been effective for solving a variety of problems, see [Davis, 1991]. However, being able to control the algorithm's parameters during evolution, not only to fine-tune the algorithm's performance but also for gaining an insight into the effects of the parameter settings, was a widely

Table 3.16: Question 10.1. The comments made by the respondents with regard to the use of a bi-directional execution control panel. The respondents could either do this already and did not comment further, considered this useful, in general or for educational or debugging purposes, or considered the use of navigational control more effective for offline visualizations than online visualizations.

BI-DIRECTIONAL CONTROL	USERS	INDIVIDUAL RESPONDENTS
<i>Can already do it</i>	1	<i>T2</i>
<i>Very Useful</i>	13	<i>T3 R6 R1 R5 R2 R4 R3 A5 A8 A2 A6 A3 A1</i>
<i>Better Done Offline</i>	2	<i>T1 A7</i>
<i>Useful for Education</i>	1	<i>R7</i>
<i>Useful for Debugging</i>	1	<i>R8</i>

accepted advantage, and one that does not exclude the use of an adaptive parameter scheme. On the contrary the use of an online parameter editor would be particularly useful for exploring the capabilities of an adaptive parameter scheme.

Editing the chromosomes either in the current population or in the reproduction gene-pool was generally considered to be a strange idea (see Tables 3.18 and 3.19), since this goes directly against the underlying principle of survival of the fittest. Under these conditions the user could genetically engineer their own solutions. Several of the respondents in all three groups considered this to be ineffective meddling. Some of the respondents in the research group noted some possible advantages as a teaching aid, and some respondents in the applications group considered it to be a useful method for seeding the algorithm with new chromosomes.

The questionnaire then invited the respondents to make any other suggestions regarding interaction that they felt would be useful (Appendix B). Seven of the nineteen respondents mentioned some additional form of interaction (theory 1/3, research 4/8, applications 2/8). These suggestions were with regard to the initialization (both as a method for seeding the initial population and re-initialization), parameter editing (specifically for introducing a strong mutation kick), and for exploration (selecting individuals to be shown at a finer level of detail).

So, although navigating through the evolution of a GA and altering the algorithm's parameters online was considered useful, caution was expressed at taking an invasive approach to editing the

Table 3.17: Question 10.2. The comments made by the respondents with regard to using an editor to change the GA's parameters during execution. The comments here are similar to those in Table 3.16, two of the respondents however considered this to be disruptive to the GA's evolutionary search.

EDIT PARAMETERS	USERS	INDIVIDUAL RESPONDENTS
<i>Can already do it</i>	1	<i>T2</i>
<i>No</i>	2	<i>T1 A5</i>
<i>Disrupts GA's Evolution</i>	2	<i>R7 R1</i>
<i>Useful</i>	9	<i>T3 R3 R5 R2 A8 A2 A6 A1 A7</i>
<i>Useful if GA Gets Stuck</i>	3	<i>R8 R6 R4</i>
<i>Useful for Education</i>	1	<i>R7</i>
<i>Useful for Experimenting</i>	1	<i>R1</i>
<i>Problem Dependent</i>	1	<i>A3</i>

chromosomes as this would detract from the underlying principle of fitness biased survival.

3.2.5 Any Other Comments

Of the nineteen respondents five made further comments (see Appendix B for details). One suggested the development of a method for estimating how long the GA would take to achieve a desired fitness rating. Another indicated a preference for supporting the design of GAs rather than visualizing their execution. The remaining three emphasized the need for a tool that was generalizable and flexible - generalizable in terms of being suitable for a range of GA applications or different EC paradigms, and flexible in terms of being easy to change and suitable for visualizing information from an individual GA run as well as from a series of runs.

3.2.6 Summary

This subsection summarizes the findings and failings of the study discussed above. The suggested visualizations included in the study were based upon the related work done in GA visualization and SV. Although the current "state of the art" was used as a foundation for the suggested visualizations, it was not expected that these would be perfect for all users or relevant to all tasks. The following

Table 3.18: Question 10.3. The comments made by the respondents with regard to using an editor to change the population's chromosomes between generations. This was considered to be a strange idea that disrupts the GA's search and in some cases should not be attempted.

EDIT POP CHROMOSOMES	USERS	INDIVIDUAL RESPONDENTS
<i>No</i>	5	<i>T1 T2 T3 A3 A1</i>
<i>Silly - Interferes with GA</i>	3	<i>R1 R3 A5</i>
<i>Useful</i>	7	<i>R5 R7 R2 R4 A2 A6 A7</i>
<i>Limited Use</i>	2	<i>R6 A8</i>
<i>Useful if GA Gets Stuck</i>	1	<i>R8</i>

key findings can be drawn from the results of the study:

1. GA users are highly computer literate, often experienced in the use of more than one computer platform and familiar with programming in more than one computer language.
2. GA users can be categorized by their primary motivation for using GAs. The respondents were categorized as either using GAs to further their understanding of GA theory through experimental testing, to explore how GAs can be applied to different problem domains, or to solve a specific problem at hand.
3. GA users examining the theory of GAs (i.e. the theory group) were interested in the specific details of their study, such as the representation used and the effects of different algorithm designs. Providing a closed set of views for the study of GA theory is impractical. New developments within GA theory define new items of interest and a need for new GA visualizations. Hence, to support these users flexible visualization support must be supplied that can be adapted to match the visualization requirements of the experiment at hand.
4. GA users exploring the application of GAs to different problem domains (i.e. the research group) experienced problems due to their knowledge of the problem domain. Little or no problems were experienced when working on familiar problems or domains that could be easily encoded and evaluated, but for other unfamiliar or complex domains difficulties were common when defining a new representation scheme, evaluation method and/or selecting new or unfamiliar

Table 3.19: Question 10.4. The comments made by the respondents with regard to using an editor to change the gene-pool's chromosomes within a generation. The comments made here are similar to those referred to in Table 3.18.

EDIT GENE-POOL CHROMOSOMES	USERS	INDIVIDUAL RESPONDENTS
<i>No</i>	8	<i>T1 T2 R5 R4 A3 A1 A7 A5</i>
<i>Disruptive for GA</i>	2	<i>R1 R7</i>
<i>N/A - Steady State GA</i>	2	<i>R6 R3</i>
<i>Maybe Useful for Education</i>	2	<i>T3 R7</i>
<i>Useful</i>	2	<i>R2 A2</i>
<i>Useful if GA Gets Stuck</i>	2	<i>R8 A6</i>
<i>Of Minor Value</i>	1	<i>A8</i>

evolutionary operators.

5. GA users working in specific problem domains, adopting GAs solely as a relevant problem solving method (i.e. the applications group), also experienced problems due to the complexity of their problem. These users had sufficient expertise to construct appropriate representations without too much difficulty. However, their problems were due to a lack of existing knowledge with regard to evaluating the problem solutions within the problem domain.
6. GA users (irrespective of their task), typically used default (tried and tested) algorithm components and parameter settings
7. GA users make improvements to their algorithms' designs through trial and error by making small changes, executing their algorithm and checking the results.
8. Overall, fewer than half of the respondents explored the quality of their algorithm's solutions and less than a third explored their algorithm's coverage of the search space, resulting in little or no link being established between the algorithm's design and its search behaviour.
9. Presenting the details of every chromosome in every population is too much data for the GA user to monitor effectively, user-defined sub-sets may miss out important information, and fitness

graphs (although useful for seeing the algorithm's results) do not provide any insight into the local structure of the fitness landscape.

10. Similarity measures between chromosomes may be used to provide an indication of population diversity but care must be taken to ensure that the similarity metrics used are appropriate for the problem domain being explored. For example, the Hamming distance between each chromosome in the population and the fittest chromosome can produce non-unique values which may confuse the user; 000000 is equi-distant from 010101, 101010, 000111, 111000, 101100, 100011, etc.
11. Navigational interaction for stepping through the generations of an evolution is a useful means for reviewing the evolutionary search path.
12. Editing the algorithm parameters can be useful for fine-tuning an algorithm and guiding its evolution.
13. Editing the contents of the algorithm's population is contrary to the underlying principles of evolution but may be a useful method for seeding the population.
14. A GA visualization environment must be sufficiently usable to allow the user to apply off-the-shelf standard visualizations as well as being sufficiently expressive to support the design and development of new visualizations.

In an attempt objectively to judge the efficacy of the study, the original study issues used to design the questionnaire are re-visited here. Like any free-response questionnaire this study attempted to provide GA users with every opportunity to express their opinions. The study was extremely effective in this respect, identifying the working practice of GA users and their opinions toward GA visualization and Human-GA interaction.

Although the questionnaire was intended to be completed within half an hour, some of the responses were so rich in the information supplied that this implicit time limit may well have been broken. Concerns over the respondents' anonymity would appear to be unfounded as the responses given were both frank and direct, however, judging the effects of the respondents' anonymity are impossible given that no other experimental conditions regarding the respondents' identity were explored.

The assumptions made regarding the nature of GA users were supported by the results of the study. The respondents were working in both industry and academia, they were located all over the world, and access to computers was not a problem for these users. In many cases several different computing platforms were available.

Identifying the respondents' background and establishing permission to contact them again in the future proved particularly valuable with regard to the two respondents developing their own toolkits. Contact with these individuals provided access to a wider range of experience and additional insight into alternative GA visualizations.

3.3 Conclusions

This section concludes this chapter with a set of typical user queries and a summary of the visualization requirements of GA users. The responses given in this study indicate that GA users are interested in seeing how their algorithms search the problem space, and not just in the results that their algorithms achieve. From this we can derive a set of users' questions, for which GA visualizations may provide some answers, specifically:

- How diverse/converged are the chromosomes in the population?
- Are there clusters of chromosomes forming during the GA's run?
- How does the local structure (i.e. schemata) of the chromosomes affect the chromosomes' fitness ratings?

In order to satisfy these typical user queries the following set of required visualization features were extracted from the questionnaire responses:

- **Usable**
 - *Ready to use generic GA visualizations.* All GA users work within the basic paradigm of the genetic evolution of chromosomes; standard frequently-used generic views, such as the fitness versus time graph, should be readily available without introducing any additional programming overheads.

- **Expressive**

- *The user should be able to introduce new visualizations.* Different users have different reasons for using GAs, such as exploring the theoretical aspects of GAs, researching the application of GAs in a specific problem domain, or practically applying GAs as a generic problem-solving method. The differences in these users' motivation correspond to differences in their use of GAs and differences in their visualization needs.
- *The user should be able to introduce problem-specific performance measures and problem-specific visualizations.* GAs are applied to a wide range of different problem domains, the user should be able to introduce problem-specific measures or problem-specific visualizations within these different problem domains to support the users' interpretation of the GA's behaviour.
- *The user should be able to reuse components of existing visualizations.* Layered support for the construction of new visualizations allows the user to reuse previous GA visualizations (e.g. the fitness rating versus time graph) or existing types of views (e.g. 2D line graphs) without removing the opportunity for the user to revert to code level graphics programming in order to express something completely new. The reuse of existing components improves the usability of the tool but this should be balanced against the freedom of expression needed to produce visualizations of algorithms and problems as yet unknown.

- **Interactive**

- *Bi-directional control for viewing the GA's execution generation by generation.* Navigational support should be available to help GA users explore their algorithms' evolutionary search behaviour.
- *Algorithm parameter and component editing.* GA users should be able to edit their algorithms' components and parameters and explore the effects that their design changes have on the behaviour of their algorithms.
- *Chromosome editing.* Editing the chromosomes in the population is one method for re-introducing genetic diversity into a converged population, or seeding a population in order

to bias its evolution toward specific regions in the search space. However, this breaks the underlying EC principle of survival of the fittest.

- **Supportive**

- *GA design support.* GA users should be supported when attempting to represent new problems, develop new evaluation functions or construct new algorithms.
- *GA search space visualization.* Support for the user's interpretation of their algorithm's behaviour should be made available wherever possible. Without understanding what an algorithm does the GA user has no way of interpreting the quality of their design or the results the GA discovers.

Chapter 4

Review of Related Work

This chapter examines the support already available for fulfilling GA users' visualization requirements. Section 4.1 describes a range of visualizations available for showing the key characteristics of GAs suggested in the overview of EC (Chapter 2). Section 4.2 presents a brief overview of some systems which exemplify these key characteristics. Finally, Section 4.3 concludes this chapter with a summary of the contributions made by these systems.

4.1 Visualizing a GA's Key Characteristics

Section 2.3 identified a set of seven key characteristics of GAs that were considered potentially useful for understanding a GA's search behaviour; namely: showing the operation of the GA, the quality of the solutions found, the chromosomes' genotypes and phenotypes, the GA's sampling of the search space, the user's ability to navigate through the GA's execution, and editing the GA's population or algorithm configuration. These were used to inform the design of the GA user questionnaire used in Chapter 3, and are used again here to structure this section, where their relevance to the GA user's visualization needs is discussed along with the visualization support currently available; each subsection concludes with a summary of the contributions made.

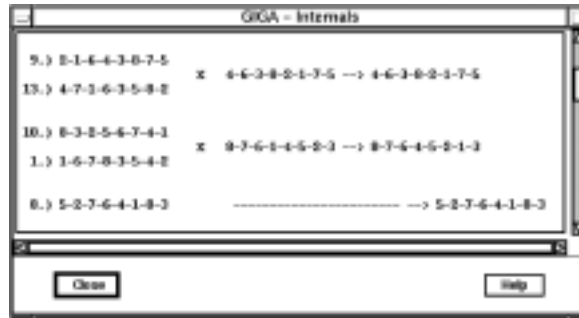


Figure 4.1: The internals window available in GIGA for showing the actions of the reproduction operators of a GA. This example was taken from [Dabs and Schoof, 1995, page 8].

4.1.1 The Operation of the GA's Component Parts

Visualizing the operation of the GA's component parts, i.e. the actions of the algorithm's selection and reproduction operators, can be done either by using a static or dynamic illustration. The static illustration benefits from being easy to present on paper as well as the computer screen, although viewing a dynamic illustration (i.e. an animation) is often a more effective and more engaging representation.

An example of a static illustration of a GA's components is the "internals window" available in the Graphical user Interface for Genetic Algorithms ("GIGA") [Dabs and Schoof, 1995]. This view illustrates the internal operations of the GA such as the crossover and mutation operators (see Figure 4.1). A sample dynamic algorithm animation of a GA was produced by David Brogan using an SV system called "TANGO" [Stasko, 1989]. Brogan's illustrative example is included in the example visualizations supplied with the X windows version of TANGO, available via ftp from per.cc.gatech.edu (see directory `/pub/xtango`). A screen view is shown in Figure 4.2 depicting both phenotype and genotype visualizations. An algorithm animation is shown at the bottom of the view which illustrates the actions of the GA's genetic operators.

Contribution

The actions of the GA's operators drive the GA's search in the problem space. Therefore, it would be reasonable to assume that illustrating the execution of the operators would provide some insight into GA's search behaviour. However, both the GIGA and TANGO visualizations of the GA's genetic

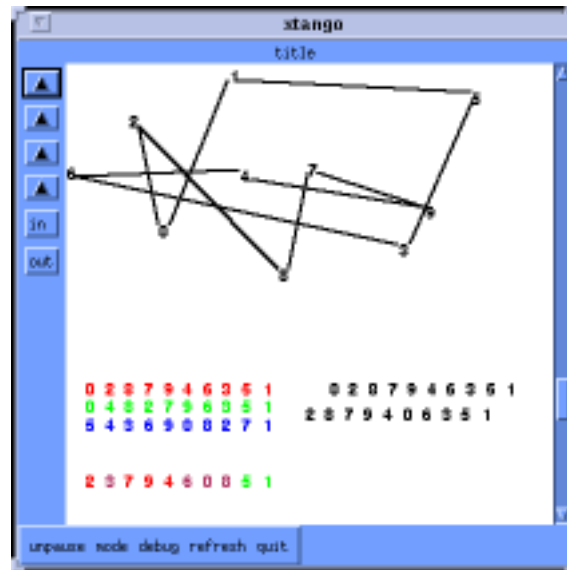


Figure 4.2: An XTANGO visualization illustrating a GA with a population containing three decimal valued chromosomes. The upper section illustrates the phenotype data i.e. the traveling salesman problem, the lower section shows an animated view of the genotype data (i.e. the decimal chromosomes) and the actions of the selection and reproduction operators used in the GA.

operators are impractical for real problems. Neither visualization scales up for use on standard-sized GA populations. Furthermore, the level of insight that can be achieved from these views is at the microscopic level of the chromosomes' genes and provides little insight into the behaviour of the overall system.

Examining the actions of the GA's selection and reproduction operators was one of the GA characteristics that the study respondents were not directly interested in other than as an educational or debugging aid. Therefore, the visualization of the GA's operators is not pursued further within this thesis, although provision for such support could be made in the future using the visualization framework presented in Chapter 6.

4.1.2 The Quality of the Solutions Found by the GA

Examining the quality of the solutions found by a GA is an important part of applying a GA. Monitoring the GA's progress can be used to inform the user's decision to end the GA's run, or as a post-mortem technique for illustrating the GA's run. This subsection presents a variety of techniques for showing the quality of a GA's solutions, including both summaries and complete accounts of the

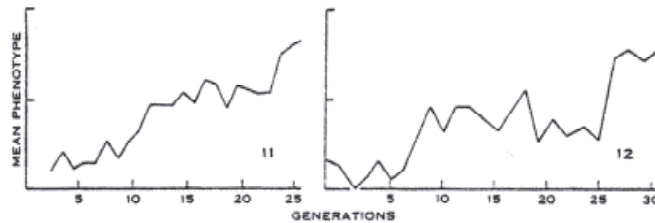


Figure 4.3: Two mean phenotype versus generation number graphs taken from [Fraser, 1957] (Figures 11 and 12).

entire run’s results, as well as the results in individual populations.

The standard method for presenting a summary of the entire run’s results is to plot some aspect of the population’s fitness ratings for each generation. These visualizations are commonly referred to as “fitness versus generation number” or “fitness versus time” graphs. Fitness versus time graphs first appeared in one of the earliest papers on simulated evolution written by A. S. Fraser in 1957 [Fraser, 1957]. Fraser used 2D line graphs to illustrate the changes in the population’s average phenotype fitness value over successive generations. An example taken from Fraser’s paper is shown in Figure 4.3.

A variety of fitness versus time graphs are commonly used today, examples include “online” and “offline” fitness ratings (i.e. the mean fitness rating, and mean current-best fitness rating across all generations [De Jong, 1980]), as well as the best and worst fitness ratings in each population [Goldberg, 1989], [Davis, 1991], [Baeck, 1996].

Although the fitness versus time graph is the most commonly used representation of the GA’s entire run, it is incomplete in that it only provides an *indication* of the chromosomes’ fitness ratings in each population rather than the *actual* chromosomes’ fitness ratings. The 3D fitness graph presented in [Harvey and Thompson, 1996] provides a more complete view, showing the fitness ratings of every chromosome in a fitness-ordered population (see Figure 4.4).

The 3D fitness graph presents all the chromosomes’ fitness ratings, but if presented as a static view some sections of the lines may be hidden by earlier and fitter line sections. A solution to this

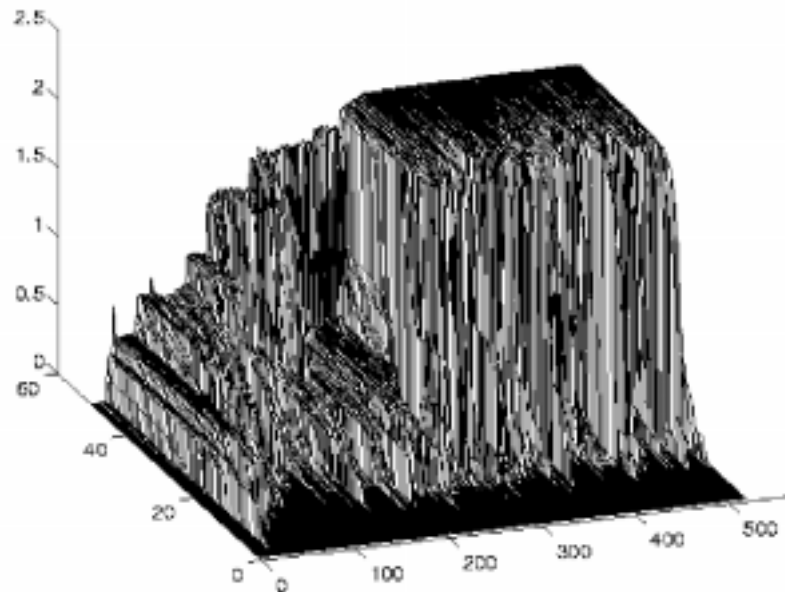


Figure 4.4: An example of a 3D fitness graph. The fitness rating of each individual in the population is plotted over each generation. The fitness ratings are plotted on the y axis ($y = 0$ to 2.5), the position of each chromosome in the fitness ordered population is plotted on the x axis ($x = 0$ to 50), and the generation number is plotted on the z axis ($z = 0$ to 522). This figure was taken from [Harvey and Thompson, 1996].

problem is to let the user control the viewing position by rotating the 3D image about its own axes. Another point to be noted regarding the 3D fitness graph is that the individual lines do not refer to the same chromosomes, rather they refer to chromosomes at the same position in the fitness ordered population across different generations.

Rather than examining the quality of the solutions found during the course of the GA's run, a number of visualization techniques for illustrating the fitness ratings of the chromosomes in a single generation were proposed in a previous project [Collins, 1993]. The techniques explored included block diagrams, colour maps, bar charts, radial line graphs and radial point plots.

"Hinton diagrams" are used in the study of artificial neural networks to illustrate the strengths of the links between the nodes in a network (see [Rumelhart and McClelland, 1986, page 103]). A Hinton diagram is made up of a series of coloured blocks used to indicate the network weights, the size of the block indicates the magnitude of each link's weighting, and the colour; black or white, indicates whether the weight is positive or negative. A diagram based on the Hinton diagram illustrates the fitness values of the chromosomes in a population (see Figure 4.5). The size of each block indicates

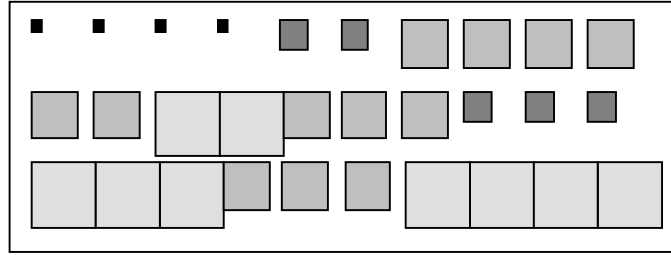


Figure 4.5: A Hinton style block diagram. This figure illustrates each chromosome in the population as a square block; the size of the block indicates the chromosome's fitness rating. Colour is used to highlight the chromosomes' level of fitness, here the fitness ratings are split into four bands corresponding to the four sizes of blocks used in the figure. This figure was taken from [Collins, 1993a] where texture was used to indicate colour on a black and white printer.

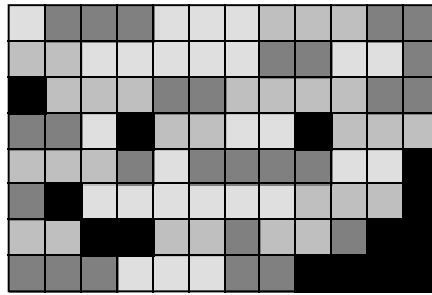


Figure 4.6: A Colour Map showing the fitness rating of every chromosome in a population, each chromosome is represented as a block; the block's colour indicates the chromosome's fitness rating. This figure was taken from [Collins, 1993a] where texture was used to indicate colour on a black and white printer.

each chromosome's fitness rating, and its colour in the spectrum red through to blue indicates the chromosome's fitness rating in the range of the minimum to maximum fitness ratings found during the GA's entire run.

A colour map shows the fitness rating of every chromosome in a population. These are similar to Hinton-style diagrams but use colour only to indicate each chromosome's fitness rating, the size of each square remains constant (see Figure 4.6). The ordering of the individual squares in a colour map can be used to illustrate different aspects of the population, ordering by fitness emphasizes the frequency of individuals with similar fitness ratings, whilst ordering by a similarity measure can emphasize the diversity of the population and the possibility of multiple solutions.

Coastline fitness diagrams show the fitness rating of each chromosome in the population as a long

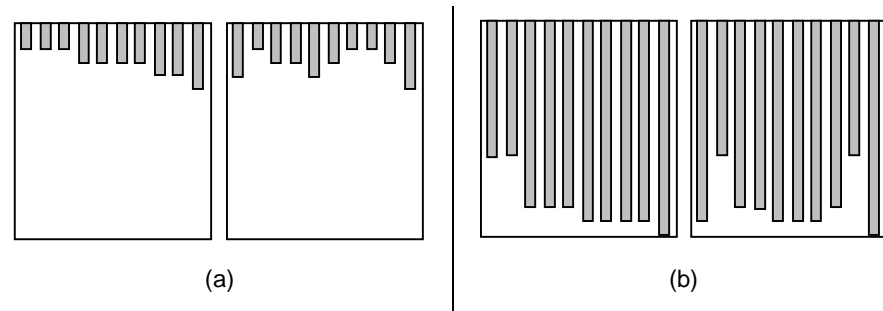


Figure 4.7: A Coastline Fitness Diagram showing the chromosomes in two populations (a) and (b), here a fitness ordered view is shown on the left and a similarity (i.e. Hamming distance) ordered view is shown on the right for populations (a) and (b). Both views are ordered from left to right for increasing fitness and similarity ratings. This figure was taken from [Collins, 1993a].

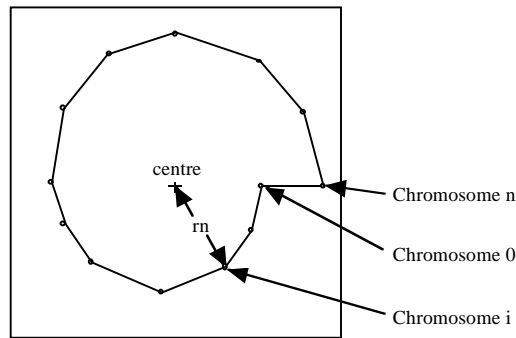


Figure 4.8: A radial plot of the fitness ratings in a single generation. The radial line trace shows the fitness ratings of the chromosomes in a fitness ordered population the distance (m) from the centre to the line indicates the magnitude of the fitness rating.

vertical bar; the height of each bar indicates each chromosome's fitness rating. Like colour maps, different ordering methods can be applied in order to illustrate different features of the population. For example, the fitness rating could be used to illustrate the diversity in fitness, or a similarity rating (such as Hamming distance to the fittest) can be used to indicate the diversity in the chromosomes' values. Figure 4.7 shows the coastline fitness diagrams of two populations, one for an unfit population (a) and one for a fit population (b), the two views in each case illustrate alternate ordering methods; by fitness (shown on the left) and Hamming distance to the fittest (shown on the right).

In a radial fitness diagram a single radial line trace is used to illustrate all of the chromosomes' fitness ratings in a fitness ordered population. The angular position indicates each individual chro-

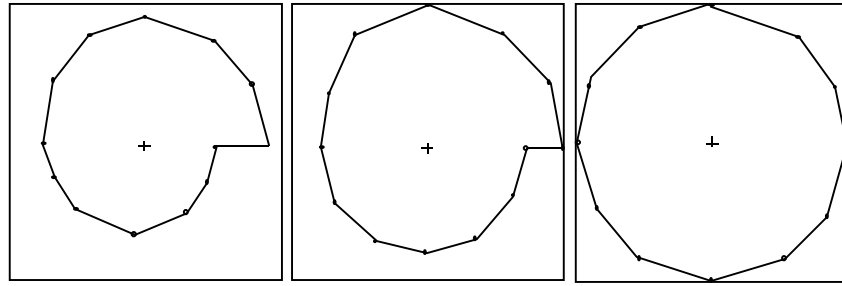


Figure 4.9: Three radial fitness plots illustrating three different stages during an algorithm's execution.

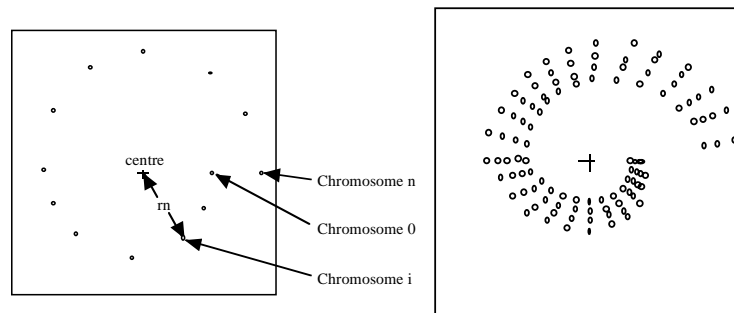


Figure 4.10: A radial fitness plot. Each individual's fitness rating is represented as a dot. The fitness ordered position of each chromosome is represented by the angular position of each dot, the distance (m) of a dot from the origin indicates the magnitude of the fitness rating, and the dot's colour indicates the generation number in which it last appeared.

mosome's position in the fitness ordered population, and the distance from the centre of the plot to the line trace indicates the magnitude of the fitness rating (see Figure 4.8). Initially the trace is a spiral, highlighting the difference between the worst and the best fitness ratings, however as the chromosomes converge their fitness ratings become similar and so the radial plot becomes more circular (as shown in Figure 4.9).

The final fitness plot suggested in [Collins, 1993] was a Fossil fitness diagram. These can be used to present either the fitness ratings of the chromosomes in a single generation, or the fitness ratings of all the chromosomes in every population across a number of generations (see Figure 4.10).

In both cases each chromosome is represented as a dot. The angular position of each dot indicates the chromosome's position in the fitness ordered population, the distance from the centre of the display to each dot indicates the chromosome's fitness rating, and the colour of the dot indicates the generation in which that chromosome last appeared, ranging from red for generation 0 to blue for the

final generation. The overall result is a series of coloured markings, similar in shape to an ammonite (i.e. a spiral fossil). The number of dots at each angular position illustrates the diversity in the chromosomes' fitness ratings, in the fitness ordered population, over an entire GA run. Again like the 3D fitness graph, the same angular position does not indicate the same chromosome in different generations, rather each angular position shows all of the chromosomes at the same position in a fitness ordered population.

Contribution

None of the fitness plots described in this subsection suffer from any scaling problems, all of these plots are applicable to any size of population and any form of EA. Table 4.1 summarizes the defining characteristics of each fitness visualization.



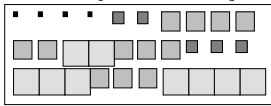
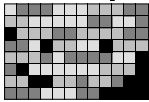
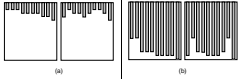
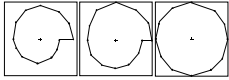

Although there are a range of visualizations available for showing the quality of the solutions found during a GA's run, the results of the GA user study indicated that the traditional 2D fitness versus time graph was by far the most popular (see Section B.3, Question 7.3). However, several respondents also indicated a need for a more detailed understanding of the GA's run. These responses refer to a need to understand the local structure of the search space and the relationship between the local structure and fitness ratings, rather than a more complete understanding of the chromosomes' fitness ratings in each population. The provision of this is discussed in subsection 4.1.5.

4.1.3 The Chromosomes' Genotypes

Viewing the chromosomes' genotypes is usually carried out either for an entire population or a subset of the population, for example by displaying the best chromosome or top five chromosomes in each generation. Although displaying the genotype of a few chromosomes per generation gives the user an indication of the solutions currently being considered it is impossible for the user to view every chromosome from every generation and grasp the GA's behaviour - there is simply too much information for the user to deal with. As a result, several systems have been developed using visualization techniques to represent this information in a more manageable form.

Three chromosome icons were introduced in [Collins, 1993] for illustrating the chromosomes' genotypes; the "trace icon," "DNA strip" and "colour strip" (see Figure 4.11). A "trace icon" is a 2D

Table 4.1: The defining features of a range of EA fitness visualizations.

VISUALIZATION	GRAPHIC	CONTENT	PERIOD
<p><i>2D Fitness vs time graphs</i></p> 	2D line graph	Summary of fitness ratings	per generation for every generation
<p><i>3D Fitness vs time graphs</i></p> 	3D line graph	Every chromosome's fitness rating	per generation for every generation
<p><i>Hinton style block diagrams</i></p> 	2D extended point plot	The chromosomes' fitness ratings	for a single generation
<p><i>Colour maps</i></p> 			
<p><i>Coastline fitness diagrams</i></p> 			
<p><i>Radial fitness diagrams</i></p> 	2D radial line graph		
<p><i>Fossil fitness diagrams</i></p> 	2D radial point plot		for a single generation, or per generation for every generation

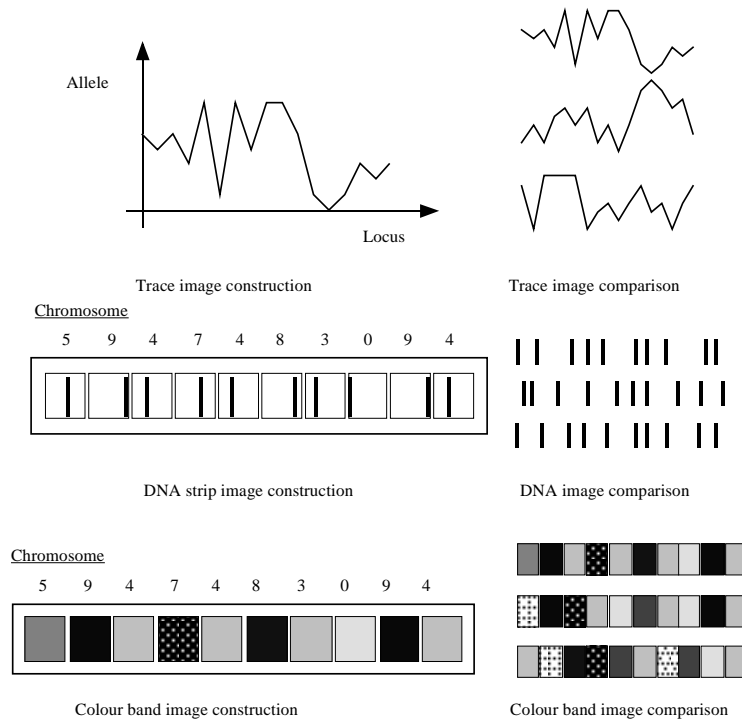


Figure 4.11: Three example chromosome icons showing the design of line trace, DNA strip, and colour band icons. This figure was taken from [Collins, 1993a], where texture was used to indicate colour on a black and white printer.

line trace of the allele held at each locus in the chromosome. The variation in the vertical position of the trace at each line segment indicates the allele’s position in the coding alphabet for each locus. A “DNA strip” is a 2D line plot showing each allele as a vertical bar, the horizontal position of the bar indicates the allele’s position in the coding alphabet. Thirdly, a “colour strip” icon shows the allele held at each chromosome locus as a coloured block, the colour of each block indicates the allele’s position in the coding alphabet.

Bill Spears at the US Naval Research Lab has also explored the use of visualization within GAs [Spears, 1994]. In order to illustrate the chromosomes in a specific population Spears suggested illustrating the alleles in a population of binary chromosomes as black and white pixel dots. The resulting pixel-oriented visualization shows a random set of black and white pixels for the initial population with patterns of vertical black and white lines forming during the GA’s run indicating common genes between neighbouring chromosomes (see Figure 4.12).

Although developed separately, the pixel-based genotype visualization proposed by Spears is similar to the color strip icon proposed by Collins. Spears’ representation can illustrate bigger genotypes



Figure 4.12: A high dimensional visualization showing a population of 100, 1008 bit binary chromosomes as black and white pixels. The entire population is shown here in 100 rows each chromosome is shown as a single row of 1008 pixels.



Figure 4.13: An example of a VIS “run window,” illustrating the best individual from each generation using a “zebra” representation.

than the color strip icon in the same amount of screen space, but the legibility of each pixel point would be poorer than the legibility of each coloured block. For any specific application the purpose of the visualization should be used to determine the balance between screen economics and image legibility. The purpose of Spears’ pixel-oriented visualization is to help people spot emerging patterns within the population, where as the purpose of the colour strip icon was to directly illustrate the alleles in each chromosome’s genotype.

Another more recent project at the US Naval Research Lab has been exploring the use of GAs for modelling viruses (the “Virtual Virus” project [Grefenstette et al., 1997]). As part of this project an offline (post-mortem) visualization tool called VIS has been developed to support the detailed analysis of a GA’s run [Wu et al., 1997], [Wu et al., 1998]. VIS presents three different perspectives on a GA’s run. *Run windows* display information on the entire run (typically showing one entry per generation, see Figure 4.13). *Population windows* display single individuals from a single generation (see Figure 4.14). Thirdly, *Individual windows* display information about a single individual (see

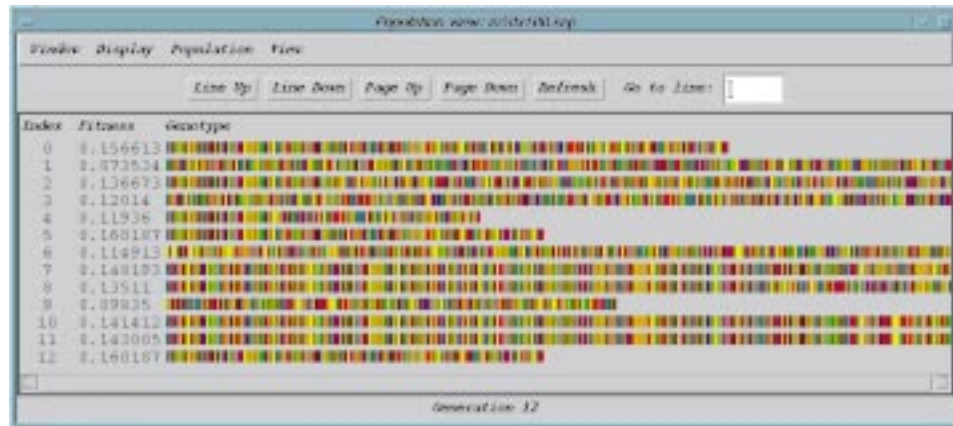


Figure 4.14: An example of a VIS “population window” illustrating all the individuals in a single generation using a “zebra” representation.

Figure 4.15). Within VIS multiple windows can be viewed simultaneously.

Five different genotype representations are available in VIS, namely; “text,” “zebra,” “neapolitan,” “colour coded” and “gene location” representations. The representation used within any of the windows can be changed at any time via the “Views” menu. The *text* representation simply displays the individuals using a fixed width type font. The *zebra* representation displays binary chromosomes as strips of black and white bars, like a zebra’s stripes. The *neapolitan* representation displays every pair of binary alleles as a coloured bar, where 00 = black, 11 = white, 01 = magenta, and 10 = orange. The *colour coded* representation is used to illustrate multi-letter alphabets (i.e. coding alphabets with more than two symbols), where each unique letter is shown by a different coloured bar (e.g. A = blue, C = red, G = yellow, and T = green). Finally, the *gene location* representation can be used to highlight the occurrence of building blocks (i.e. groups of symbols or partial solutions), different coloured strips are used to identify different building blocks.

Although it is easier to identify trends within the population using a chromosome icon representation rather than printed text, both printed text and chromosome icons present the same amount of information and therefore, suffer from the same drawback i.e. when applied to large populations they both contain too much information for the user to deal with. As a solution to this [Collins, 1993] proposed three composite representations for summarizing the chromosomes’ genotypes; “overlaid line trace icons,” “population bar charts,” and “allele versus locus frequency matrices” (see Figure 4.16).

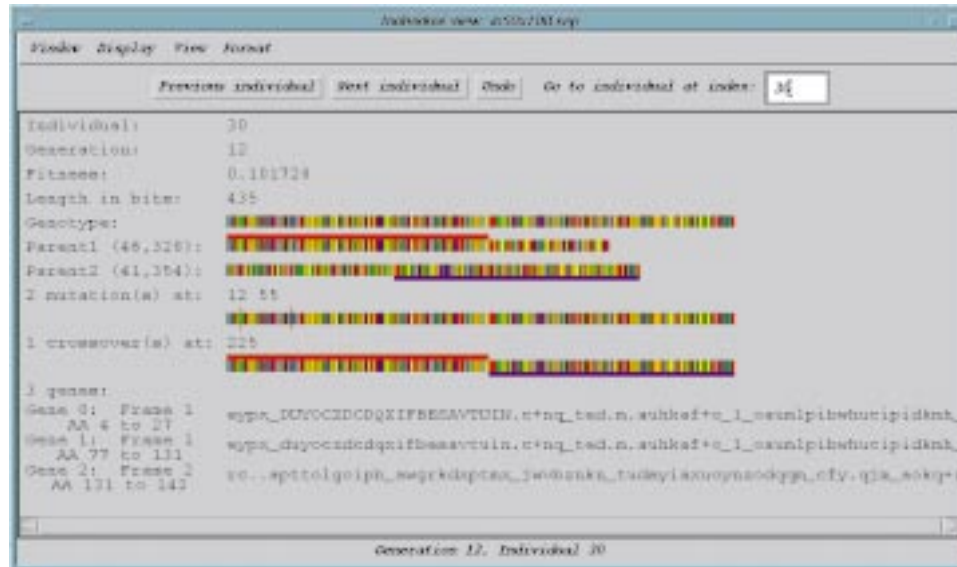


Figure 4.15: An example of a VIS “individual window” showing the data held on a single individual.

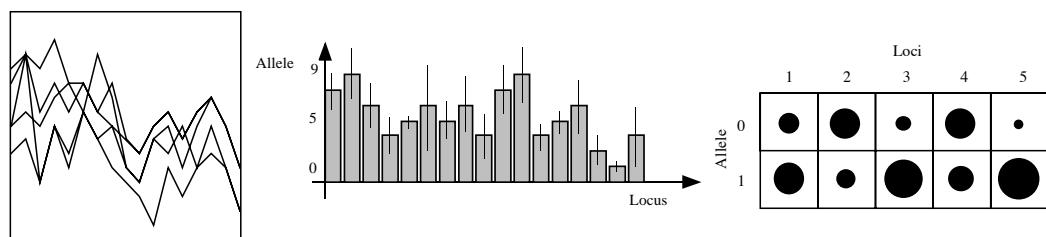


Figure 4.16: Three example genotype visualizations; “overlaid chromosome icons” (left), a “population bar chart” (middle) and an “allele versus locus frequency matrix” (right). These three chromosome visualizations are taken from [Collins, 1993a].

The *overlaid line trace icons* representation is produced by plotting an enlarged version of every chromosome’s line trace icon on the same set of axes. The composite image indicates the allele diversity at each locus within the population, by the number of vertically aligned separate line segments. For large (i.e. most practical) population sizes the overlaid chromosome icon representation becomes overloaded and difficult to read (see Section 5.2.3, Figure 5.4 on page 142 regarding the graphic density and angular separation of legible images). Although the line trace icons identify each chromosome and its alleles, they do not indicate the frequency of each chromosome (or chromosome building block). Therefore, the user *can* see when the population is completely converged at a specific locus, but they *cannot* see the diversity of the population prior to that point. For example, a population containing equal numbers of two different chromosomes would look the same as a population

that contained 90% of one chromosome and 10% of an other.

The *population bar chart* summarizes the alleles that are present within the chromosomes in the population. Each bar indicates the alleles present at each locus, the height of the bar is used to indicate the most frequent allele at that locus. Lines are added to indicate the minimum and maximum allele values at each locus for the current population. Although this gives an indication of the population's diversity, like the overlaid chromosome icon representation it does not illustrate the distribution of the alleles. As a result, the user is no better informed about the diversity of the chromosomes in the population.

Thirdly, *allele versus locus frequency matrices* illustrate the distribution of the allele within a population. By viewing the allele versus locus frequency matrices of subsequent generations the user can see how the allele's distribution varies during the GA's run. This shows both the convergence and diversity of the alleles. However, it does not show any information regarding the local structure of the alleles within each chromosome. The allele versus locus frequency matrix gives a clear summary of the distribution of alleles and is perhaps the clearest of the three genotype summary representations proposed in [Collins, 1993].

Contribution

Although exploring the fine-grained details of the individual chromosomes can be very useful for examining the solutions found, like the visualization of the genetic operators, it is at too fine-grained a level of detail to help people follow the overall search behaviour of the algorithm. The responses given in the GA user study (Section B.3, Questions 7.1, 7.2, and 8.1) indicated that the respondents also believed that displaying all the chromosomes would present too much information for their purpose. They also considered the selection of a subset of the chromosomes in each generation to be a difficult task, resulting in an un-representative, or possibly misleading, visualization. Therefore, genotype visualizations must be used carefully to complement the user's exploration of the the GA's search behaviour. Perhaps if used in tandem with a visualization of the GA's sampling of the search space, the fine-grained focus that genotype visualization provides could be directed toward the more significant and interesting chromosomes within the GA's run.

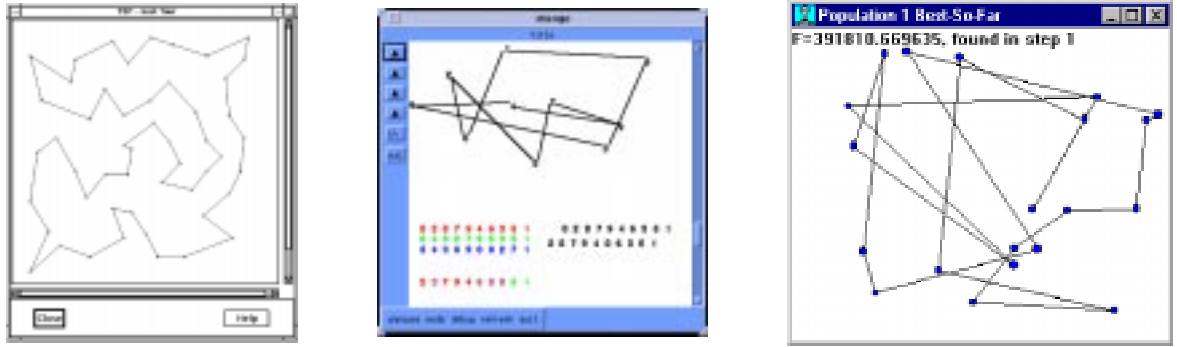


Figure 4.17: Three example phenotype visualizations for the traveling salesperson problem. These images were taken from GIGA, XTANGO and EvoNet’s Genetic Algorithm Software Development Package, respectively.

4.1.4 The Chromosomes’ Phenotypes

Visualizing the chromosomes’ phenotypes is a very effective way of illustrating the solutions being considered by the GA (see Figure 4.17). Several education-oriented GA tools illustrate the GA’s phenotypes, specifically for the traveling salesperson problem¹. Examples include the “best individual window” in GIGA, the phenotype view presented in the XTANGO sample GA visualization and the “Best-So-Far” window available in the Genetic Algorithm Software Development Package produced by EvoNet, the European Network of Excellence on Evolutionary Computation (available from <http://www.dcs.napier.ac.uk/evonet/Coordinator/html/software.html>).

Although visualizing the chromosomes’ phenotypes can produce a very salient illustration of the solutions being considered they are specific to the problem being solved and therefore, as new problems are attempted new views must be produced. If the effort involved in producing the view is perceived to be greater than the benefit achieved through its use then the user will be disinclined to produce new views.

This “ease of production” threshold is a serious problem for SV. Producing any new visualization requires some form of programming. The important issue here is to ensure that the programming involved is sufficient to fully express what the user needs, whilst remaining at a sufficient level of abstraction such that the user does not get deterred by technically demanding graphics programming.

One of the primary goals of producing an SV development environment, such as

¹The traveling salesperson problem is a problem in which the GA attempts to find the shortest route linking a set of cities.

BALSA [Brown and Sedgewick, 1985], TANGO [Stasko, 1990], ZEUS [Brown, 1991], or VIZ [Domingue et al., 1992], is to facilitate the development of new views. Although a great deal of work has been done in SV, establishing a sufficient level of expressiveness whilst maintaining ease of use is a difficult trade off (see [Repenning and Ambach, 1996]). As a solution to this problem John Stasko, author of the TANGO and POLKA SV environments, developed “SAMBA,” an interpreted, interactive animation front-end to POLKA [Stasko, 1996]. SAMBA is used by students in an undergraduate algorithms class at the Georgia Institute of Technology to produce algorithm animations from recorded data files or the output of a program piped directly to SAMBA².

Contribution

Producing problem-specific visualizations of the chromosomes’ phenotypes is a very salient illustration of the GA’s solutions. Such views explicitly illustrate the link between the chromosomes’ genotype and phenotype. This is why phenotype visualizations are so useful when illustrating the GA’s operation within an educational context. However, visualizing all of the chromosomes’ phenotypes in a typical GA produces too much information for the user to digest easily, yet like the genotype visualization described in the previous subsection selecting a representative subset can be problematic. Again, perhaps such detailed views are best used selectively to illustrate the more important chromosomes in the GA’s run.

4.1.5 The GA’s Sampling of the Search Space

The term “search space” is used repeatedly in this thesis to refer to the complete set of all allele combinations available within any given coding alphabet. Exploring the GA’s sampling (i.e. searching) of that space is one way of viewing the GA’s behaviour. This subsection describes some of the available visualizations.

In addition to his genotype visualization tool, Bill Spears also produced two visualization tools to illustrate the GA’s sampling of the search space; one for one-dimensional problems and a second for two-dimensional problems [Spears, 1994]. The first tool uses a 2D line graph to illustrate the fitness rating (plotted on the y axis) of each chromosome (plotted on the x axis). The second tool adopts

²The term “piped” is used here with reference to the UNIX pipe command “[]” e.g. “% yourprog | samba”.

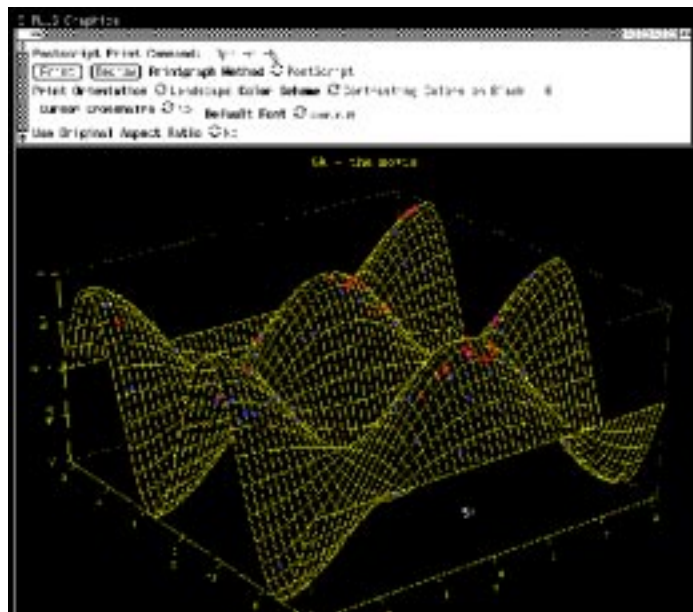


Figure 4.18: A 3D surface plot showing the fitness surface for a two dimensional search space. The chromosomes from old generations shown as blue dots and the chromosomes in the current generation shown as red dots, this figure was taken from [Spears, 1994].

a similar approach but uses a 3D plot to show the variation in fitness for two-dimensional fitness functions. In the 3D visualization the individual chromosomes are shown as points on a 3D fitness surface, as the GA evolves old chromosomes from previous generations are drawn as blue points and chromosomes created in the current generation are drawn as red dots (see Figure 4.18). Both of these tools illustrate the GA's sampling of the search space by explicitly plotting a line or surface showing the fitness landscape (i.e. the complete search space with its associated fitness ratings) and highlighting the population's sampling points. However, this approach is not possible for real problems in which the fitness landscape (i.e. the fitness rating for every possible chromosome) is unknown.

Around the same time Nassersharif, Eence and Au from the University of Nevada, Las Vegas were working on another 3D visualization of a GA's fitness landscape [Nassersharif et al., 1994]. As with Spears' second tool, Nassersharif et al. visualized GAs solving two-dimensional problems. In this case the problem space is plotted as a three-dimensional scatterplot in which the two problem dimensions are plotted on the x and z axes, with the corresponding fitness ratings plotted on the y axis (Figure

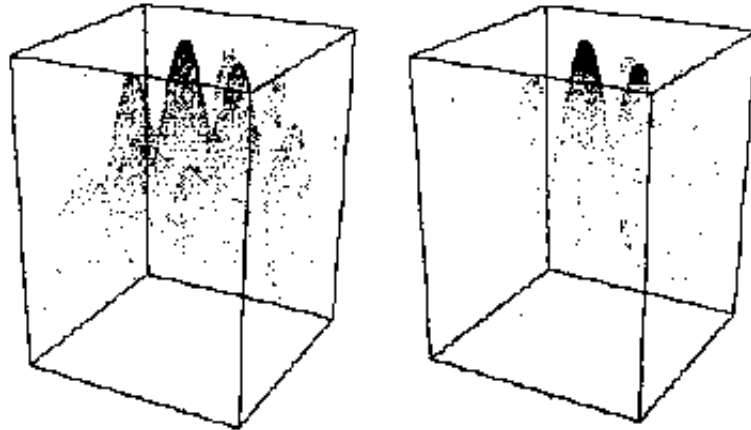


Figure 4.19: Nassersharif, Ence and Au's scatterplot visualization for GAs solving two-dimensional problems. This figure (taken from [Nassersharif et al., 1994, page 1-564]) shows scatterplots for generation 0 (left) and generation 10 (right). The x and z axes illustrate the two problem dimensions and the vertical y axis illustrates the fitness ratings, note the convergence toward fitter solutions shown in generation 10.

4.19). Rather than illustrating the entire fitness surface and then highlighting the GA's sampling of it, Nassersharif et al. used 3D scatterplot visualizations to show only the population's sample points i.e. the population's chromosomes without the fitness surface.

As noted in both [Spears, 1994] and [Nassersharif et al., 1994], GAs are not typically applied to one or two dimensional problems, they are more often applied to high-dimensional problems whose search space cannot be directly illustrated in two or three dimensional space. Therefore, a number of people have explored similarity metrics for illustrating the GA's sampling of high dimensional search spaces.

In [Collins, 1993] a suggestion was made to use 2D scatterplots to illustrate the distribution of a population's chromosomes. Each chromosome in the population can be represented by a dot in a 2D scatterplot, the coordinate of each dot indicates some problem-specific data measure, for example the chromosome's fitness rating versus its similarity measure (such as the chromosome's Hamming distance to the fittest). Selecting an informative similarity measure is the key to this view's effectiveness. As noted by several of the respondents, Hamming distance is not a very effective similarity measure. It is in fact a non-unique measure (i.e. 0000 is equidistant from 1100 and 0011). In addition to using a dot to illustrate each chromosome [Collins, 1993] also used chromosome icons to represent each chromosome, an example visualization is given in Figure 4.20.

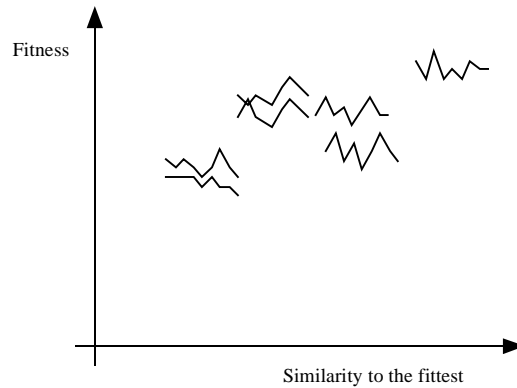


Figure 4.20: A data space view using the chromosomes fitness rating (y axis) and similarity to the fittest chromosome (x axis) to plot line trace icons of each chromosome. This figure was taken from [Collins, 1993a].

Since the time the above visualization was first proposed, further work on GA similarity metrics has been carried out as a means for judging the problem complexity and population diversity. For example, Terry Jones and Stephanie Forrest have explored the correlation between the fitness values of all the chromosomes in a GA's run and the chromosomes' similarity to the final solution (measured either by the Hamming distance for binary chromosomes or the Euclidean distance for non-binary chromosomes). The resulting measure of problem complexity is referred to as the "fitness distance correlation" [Jones and Forrest, 1995]. Simon Ronald's work on distance functions for order-based encodings (as used for representing the traveling salesperson problem) measures the genotypic or phenotypic similarity between the chromosomes in a population. These measures are then used as a means for preserving the population's diversity during a GA's run (see [Ronald, 1995], or [Ronald, 1997] and [Ronald, 1998]).

Contribution

The working practices of the surveyed GA users indicated a strong interest in the GA's sampling of the search space. When asked about visualizing the rate of change in the populations' fitness values, six of the nineteen respondents indicated that they wanted to know more about the solutions considered by the GA than the fitness versus time graph could give (see Section B.3, Question 7.3). Furthermore, the respondents were strongly in favour of visualizations illustrating a similarity rating for each chromosome in the population, such as the Hamming distance to the fittest chromosome (see Section B.3, Question 8.4). The only doubts expressed were with regard to the quality of the

similarity measure used. The choice of similarity measure was generally considered to depend on the specific problem domain and representation used in the GA.

Showing a 2D or 3D visual representation of the search space enables the user to judge the diversity of the population and identify the formation of chromosome clusters. Although a similar impression can be gained from similarity measures of the population's chromosomes, measures based on a specific search space sample (i.e. the chromosomes in a specific population or GA run) rather than the complete search space lack a consistent scale and therefore, comparisons across different populations or different runs can be difficult. However, if a consistent representation for high-dimensional problem spaces could be found then salient search space visualizations such as those proposed by [Spears, 1994] and [Nassersharif et al., 1994] could be produced for GA's solving high-dimensional problems.

4.1.6 Navigating the GA's Search

Navigating a GA's execution

GAMETER [Kapsalis et al., 1993], GIGA [Dabs and Schoof, 1995], and the Genetic Algorithm Software Development Package produced by EvoNet, are just three example systems that enable the user to "play" the GA's run like a movie, "pause" the execution of the GA, and "step" forward a single step (i.e. one generation). Using these controls the user can pause the execution of their algorithm, make a change to the algorithm's parameters and restart it, or step forward generation by generation in order to examine the GA's execution.

Within the field of SV a number of systems support the bi-directional control of the program's execution. These first appeared in systems like Henry Lieberman's "ZSTEP" system [Lieberman, 1984], Marc Eisenstadt and Mike Brayshaw's Transparent Prolog Machine ("TPM") [Eisenstadt and Brayshaw, 1987], and Thomas Moher's PROcess Visualization and Debugging Environment ("PROVIDE") [Moher, 1988]. Bi-directional navigational control over the program's execution is usually achieved by periodically recording the program's current state and then producing the visualizations using the recorded history of events. As a result, the user can navigate forwards and backwards through the program's recorded history and the resulting visualizations will show the forwards and backwards execution of the program.

Navigating a GA's Fitness Landscape

Another form of navigation that may prove useful within EC is the navigation of the fitness surface. Although generally used to navigate a program's execution a similar approach could be used to identify regions of interest within the range of fitness values from a GA's run, e.g. to identify the best chromosomes found by the GA. The navigation of the GA's execution and the discovered regions of the fitness landscape both require immediate visual feedback.

"Dynamic Queries" [Shneiderman, 1994] incorporate the use of direct manipulation and immediate feedback to query databases. An "AlphaSlider" [Osada et al., 1993] is an example of a dynamic query interface. The AlphaSlider enables users to select an item or range of items of interest within a dataset. A range-defining alphaslider looks like a regular scroll bar, except that rather than identifying a single point in a range as a small square, the alphaslider identifies a range within a range as a bar with draggable arrow buttons at both ends. These arrow buttons define the start and end of the range of interest within an ordered data set. The rectangular bar itself can also be dragged to pan across the data set. Continuous feedback keeps the user informed of their current position within the data set.

Contribution

Within the GA user study the proposal for a bi-directional control mechanism was strongly supported (see Section B.4, Question 10.1). Using a similar approach to that commonly applied within SV, a bi-directional navigation controller could be introduced for the user to navigate the GA's run, generation by generation. In addition to a movie-player styled controller for exploring the GA's execution by generation, alphasliders can be used to define ranges of fitness ratings and generation numbers to be displayed. For example, an alphaslider could be used to control the displayed content of a search space visualization, displaying the top 5% of all the generations chromosomes would show the user how many good solutions the GA had considered during its run.

Within this project GA users appear to consider their algorithms in two ways; as a series of evolving generations and as a search technique for exploring problem spaces. By enabling GA users to query a GA's execution in terms of its generation-based execution and its exploration of the problem space, both forms of understanding can be supported.

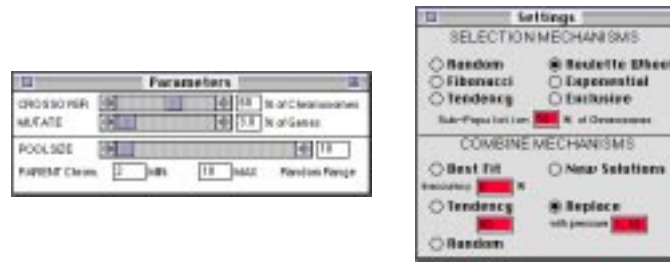


Figure 4.21: The dialog boxes available in GAMETER for editing the GA's parameter (left) and algorithm settings (right).



Figure 4.22: The bit string editor dialog used in GAMETER to edit a chromosome's alleles.

4.1.7 Editing the GA

A variety of GA systems enable the user to design their GAs using a library of predefined selection and reproduction components with default parameter settings and interactive parameter controls (examples include GAMETER [Kapsalis et al., 1993] shown in Figure 4.21, GIGA [Dabs and Schoof, 1995], EVOS [Baack, 1996], and EvoNet's GA Software Development Package). Within environments that allow the user to pause and restart the GA's execution, these settings can be altered during the course of the GA's run.

Editing can also be carried out at the data level (rather than the algorithm level). The GA user could directly alter the values of the chromosomes within the GA's population. GAMETER [Kapsalis et al., 1993], facilitates this with the use of a "Bit String Editor" (see Figure 4.22). This allows the user to edit a selected chromosome from the current population, and either set all of the alleles in a selected section of the chromosome to one, set all the alleles in a selected section to zero, or invert the alleles in a selected section (i.e. 0s to 1s and 1s to 0s). An on-the-spot evaluation can also be carried out to identify the effect of any changes made.

Contribution

Enabling the user to intervene in the evolutionary process, either to alter the algorithm or the population data, has both pros and cons. One of the pros is that providing there is sufficient visualization support, the user can explore the effects of any changes they make. This can be an engaging way to learn about the GA's search behaviour and the impact of the user's choice of algorithm design. Another pro is the fact that the user can introduce domain knowledge by seeding or biasing the population with specific genes. However, one of the cons is that any form of intervention interferes with the GA's evolutionary search. Both the pros in terms of knowledge injection and education, as well as the cons were noted by some of the questionnaire respondents (see Section B.4, Question 10.3).

The common means of altering a GA's algorithm components or parameter settings through a pop-up dialog is a clear and effective approach. However, the means for altering the individual chromosomes in a population is perhaps less obvious. The bit string editor in GAMETER allows the user to change the alleles in selected sections of a chromosome. If the aim of altering the GA's chromosomes is to introduce domain knowledge then the user must translate that knowledge into the chromosome's representation and alter the values accordingly. Yet, in practice biasing the GA's search may not be so simple as encoding a desired solution, rather the user may want to bias the GA away from sub-optimal clusters and towards unconsidered regions of the search space. Viewing the GA's sampling of the search space during the GA's evolutionary search may be one way of guiding such a choice. Within such a view it may also be possible directly to manipulate the GA's chromosome representations such that the GA is dragged away from sub-optimal clusters and toward unconsidered regions of the search space.

4.2 An Overview of the Existing Visualization Support

This section presents a brief description of each of the visualization systems referred to in the previous section. The intention of this subsection is to give the reader an appreciation of the contribution made by each system "as a whole." Subsection 4.2.1 describes each of the GA visualizations, subsection 4.2.2 describes the SV systems, and subsection 4.2.3 describes the use of information visualizations.

This work is presented in chronological order.

4.2.1 GA Systems

This subsection presents an overview of the GA systems referred to in Section 4.1.

Collins - GA VISUALIZATIONS

The proposal for this thesis was based on the work presented in [Collins, 1993], some of which is summarized in [Routen and Collins, 1993]. This earlier research identified a range of graphical representations for producing GA visualizations. A number of graphical representations were developed for showing the fitness ratings of the chromosomes in the population for a specific generation, and for showing a summary of the population's fitness ratings over a number of generations (see Table 4.1 on page 92 for a summary). Three icon representations for illustrating a GA's chromosomes were also developed, see Figure 4.16 on page 96.

Three genotype visualizations were also produced: overlaid chromosome icons, population bar charts, and allele versus locus frequency matrices (see Figure 4.16 on page 96). The last representation proposed in [Collins, 1993] is referred to as a "data space diagram" (see Figure 4.20). Each chromosome is plotted as a point on a 2D scatterplot, the chromosomes' similarity to the fittest chromosome are plotted on the x axis and the chromosomes' fitness ratings are plotted on the y axis. In addition to plotting each chromosome as a point, any of the three proposed chromosome icons can be used as point images in the data space diagram.

Representations similar to those presented in [Collins, 1993] have since been used in a range of GA visualization tools, see [Spears, 1994], [Wu et al., 1998], and [Pohlheim, 1998]. Further information on this work can be found in [Collins, 1993] and [Routen and Collins, 1993].

Kapsalis, Smith and Mann - GAMETER

GAMETER is a graphical tool that can be used on Macintosh personal computers and UNIX based workstations. Three different output windows are available showing statistical data of the GA's progress, the chromosomes in the current population and a graph of the GA's results. A bit string editor is also available for changing individual chromosomes in the population (see Figure 4.14). New

problems are introduced to GAMETER by using “skeleton files” (i.e. program templates) so that GAMETER can access the information it requires. The user can continuously manipulate the GA’s parameters and algorithm settings during the GA’s run (as shown in Figure 4.21) as well as stop, step, start and reset the GA’s execution.

Further information on GAMETER can be found in [Kapsalis et al., 1993], [Mann, 1994], or on the world wide web, see

<http://www.sys.uea.ac.uk/Research/researchareas/MAG/GAmeter/>

Spears - 5 GA Visualization tools

Bill Spears from the US Naval Research Labs in Washington, D.C. presented 5 visualization tools for exploring GAs. The first tool (referred to above as a “2D fitness landscape”) was intended for use on one variable fitness functions and presents a 2D line graph showing how the fitness rating (on one axis) varies with different variable values (on the other axis). The second tool adopted a similar approach but used a 2D surface plot to show the variation in fitness for two variable fitness functions (this has been referred to as a “3D fitness landscape,” see Figure 4.4). Spears’ third visualization tool, referred to above as a “pixel oriented visualization,” shows the binary chromosomes in a population as black and white pixels where a black pixel indicates a 1 and a white pixel indicates a 0 (an example is given in Figure 4.12, page 94).

The fourth tool, not discussed in the previous section, shows how the 2nd order schemata, i.e. two digit building blocks - 00, 01, 10 and 11, are distributed within a population. Figure 4.23 shows an example, the four triangular views show the frequency of each 2nd order schema (00 top left, 01 top right, 10 bottom left, 11 bottom right) between each pair of bits, i and j , along the chromosomes in the population. The value (i.e. greyscale) of each filled circle at position (i, j) indicates the frequency of the schema for that pair of bits. This can also be extended to show the distinction of third order schemata (i.e. 000, 001, 010 . . . 111) using eight triangular images rather than four.

Spears’ fifth and final visualization tool shows the ancestry of a GA’s population by colouring each unique individual in the initial population a different colour and then showing the chromosomes in subsequent generations as strips of colours made up from their parents. For example, if single point crossover was applied to a blue chromosome and a red chromosome, two new chromosomes would be

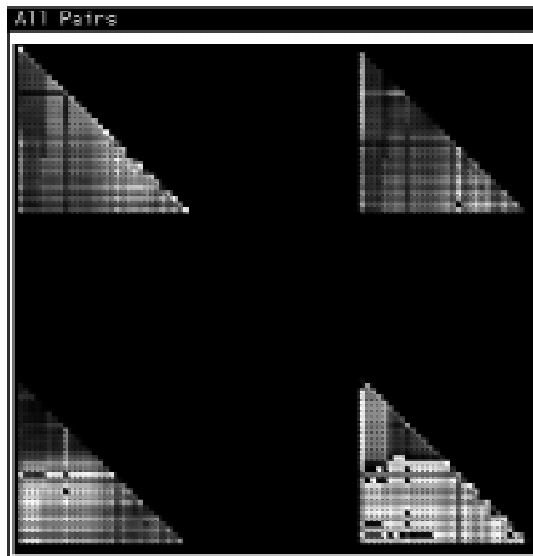


Figure 4.23: A visualization of 2nd order schemata (i.e. schemata with two defined values). The four triangular images illustrate the frequency of four different 2nd order schema across all possible combinations of loci; 00 (top left), 01 (top right), 10 (bottom left), and 11 (bottom right). The frequency of each schema is indicated by the corresponding circle's gray value; a black circle indicates that the schema does not occur in the population through to a white circle which indicates the schema appears several times.

produced - one would be shown with a red and then blue strip, and the other with a blue and then red strip. Figure 4.24 shows a population of one hundred thirty bit chromosomes after twenty five generations.

Further information on Spears' five visualization tools can be found in [Spears, 1994].

Dabs and Schoof - GIGA

GIGA is a Graphical user Interface for Genetic Algorithms aimed at providing a similar environment to that of GAMETER, i.e. an easy to use, extendable GA tool [Dabs and Schoof, 1995]. The main interface in GIGA (see Figure 4.25) provides similar functionality to the parameter and algorithm settings dialogs in GAMETER (Figure 4.21). Here the user can select their genetic operators and parameter settings within the one dialog, as well as controlling the execution, defining the termination conditions, selecting a view, and recording the GA's execution.

Execution control is possible only in the forward direction with start, pause and single step options (see Figure 4.25, bottom right). The termination conditions are set either to a particular generation

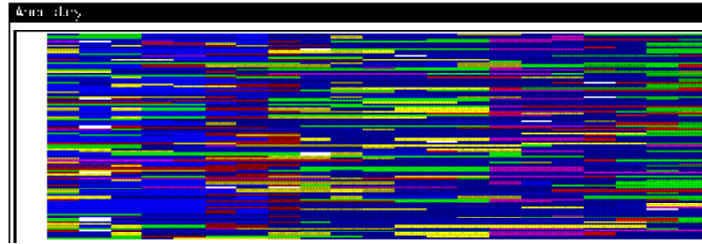


Figure 4.24: A visualization of the ancestry of a GA's population. The one hundred thirty bit chromosomes in the initial population were each given a separate colour, this visualization shows how the chromosomes from the initial generation have been recombined in order to produce the twenty fifth generation.

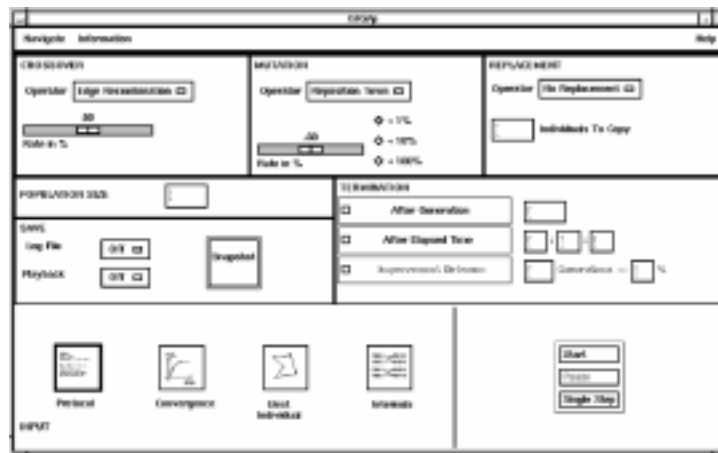


Figure 4.25: The main interface used in GIGA. Included in the interface are dialogs to alter the GA's parameters (top), a dialog to select views (bottom left), and a dialog to start, pause and step the GA's execution (bottom right). This figure was taken from [Dabs and Schoof, 1995, page 4].

number, a specific time period, or after a period of no significant improvement (see Figure 4.25, middle right). Four standard view types are available from the main interface; the *protocol* window details the GA's best individuals over the last fifteen generations, the *convergence* window presents a fitness versus time graph for either the best, average or worst individual in each population, the *best individual* window provides a phenotype visualization (see Figure 4.17) based on a (user-supplied) problem-specific view, and the *internals* window illustrates the internal operations of the GA such as the crossover and mutation operators (Figure 4.1).

A GA's execution can be recorded as either a single snapshot, playback file, or log file (see Figure 4.25, middle left). A single snapshot stores only one generation's data. A playback file stores only enough information to reconstruct the GA's execution i.e. the GA's initial parameters, the initial

random number seed, the problem data, and any parameter changes made during execution. The log file records all the information created during a GA's execution including every population's contents and parameter changes. This is intended for use after execution as a source file for further analysis or visualization. Extensions to GIGA are made by the use of template program files that can be rewritten by the user to represent their problems and algorithm components. Although this is not trivial the use of consistent, modular code packages makes this process a routine formality for those fluent in the implementation language (in this case, C).

Current work on GIGA is aimed at producing a system suitable for parallel GAs. Further information on GIGA can be obtained from Jochen Schoof (email schoof@informatik.uni-wuerzburg.de) or on the world wide web, see

http://www-info2.informatik.uni-wuerzburg.de/ga_pap_e.html

Wu - VIS

VIS is an offline (post-mortem) visualization tool to support the detailed analysis of a GA's run. The user supplies a data file of their GA's output and then applies VIS to produce textual and graphical views. The GA's execution can be explored using a bi-directional control panel. The three different views available in VIS provide three different levels of detail; *run windows* show a coarse-grained view of the GA's entire run (typically showing one entry per generation, see Figure 4.13 on page 94), *population windows* show a medium-grained view of the individuals from a single generation (Figure 4.14), and *individual windows* provide a fine-grained view of single individuals (see Figure 4.15). As stated earlier, five different representations are available for displaying the genotypes in these three views: *Text* representations simply display the individuals using text in a fixed width font. *Zebra* representations display binary chromosomes as strips of black and white bars. *Neapolitan* representations display every pair of binary alleles as a coloured bar, where 00 = black, 11 = white, 01 = magenta, and 10 = orange. *Colour coded* representations illustrate multi-letter alphabets (i.e. coding alphabets with more than two symbols), where each unique letter is shown a by a different coloured bar (e.g. A = blue, C = red, G = yellow, and T = green). Finally, the *gene location* representation illustrates the occurrence of building blocks (i.e. groups of symbols or partial solutions), where different coloured strips are used to identify different building blocks.

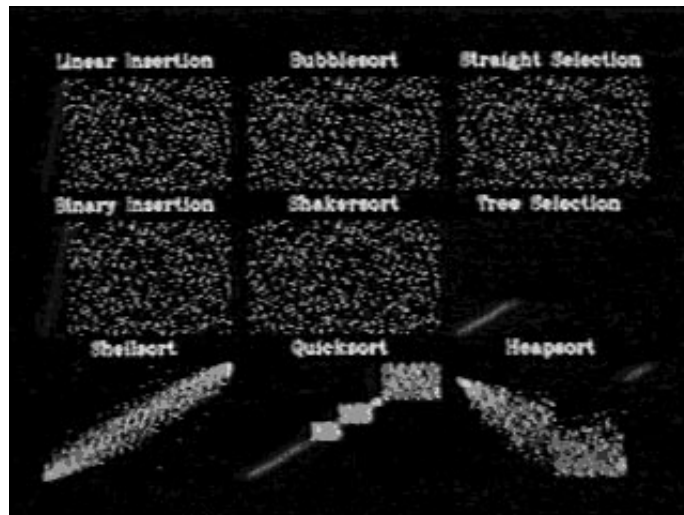


Figure 4.26: A screen image taken from Ronald Baecker's 30 minute movie on sorting algorithms. This image shows three insertion sort algorithms (left column), three exchange sort algorithms (middle column), and three selection sort algorithms (right column).

Further information on VIS can be found in [Wu and Lindsay, 1997] and [Wu et al., 1998].

4.2.2 SV Systems

The genesis of modern SV is attributed to a 30 minute narrated colour video made in 1981 by Ron Baecker at the University of Toronto [Baecker, 1981]. The video was produced in order to help people understand the operation of sorting algorithms. Baecker and his colleagues wrote a computer program that displayed the current state of a number sorting algorithm as a set of dots on the computer screen. The position of each dot indicated each number's current position in the set. A video recorder was then used to film every state of the number set displayed on the screen as the algorithm stepped through each stage of the sorting algorithm, filming a few frames of each state.

Once the algorithm had completed sorting the numbers, the video was then replayed from start to end and the dots appeared to move into place according to the behaviour of the algorithm. This approach was used to create animations of nine types of sorting algorithms: three insertion sort algorithms; linear insertion, binary insertion and shell sort, three exchange sort algorithms; bubble sort, tree sort and quicksort, and three selection sort algorithms; straight selection, tree selection and heap sort selection (see Figure 4.26).

```
(DEFUN SPLIT (5 '(4 9 1) NIL NIL)
  (IF NIL
    (LIST LESSER PIVOT GREATER)
    (IF T
      '((1 4) 5 (9))
      (SPLIT PIVOT
        (CDR LIST))
```

Figure 4.27: A screen shot taken from ZSTEP. In this code view the current focus of the stepper is shown in yellow and the recently substituted variable values are shown in red.

Since 1981 computer technology has advanced significantly, enabling the production of smooth computer animations. Although the original work done by Baecker was to support peoples' understanding of sorting algorithms, SV work is carried out on all aspects of computer software i.e. the program's code, data and algorithm. This subsection gives a brief description of the SV systems referred to in the previous section. A more complete review of SV can be found in [Price et al., 1993], [Roman and Cox, 1993], [Collins, 1995] or [Stasko et al., 1998].

Lieberman - ZSTEP

ZSTEP is a debugging tool for Lisp that integrates a code stepper with a text editor [Lieberman, 1984]. ZSTEP enables the user to follow the execution of a Lisp program by substituting values for variables in the source code during the program's execution. The user can navigate either forwards or backwards through the program's execution and "zoom in" on a bug, examining the program initially at a coarse level of detail, then at increasingly finer levels until the bug is located (see Figure 4.27).

ZSTEP94 is a more recent version of ZSTEP recently developed by Henry Lieberman, see [Lieberman and Fry, 1995] for details. Further information on ZSTEP and ZSTEP94 can be found on the world wide web, see

<http://lieber.www.media.mit.edu/people/lieber/Lieberary/ZStep/ZStep.html>

Moher - PROVIDE

The primary goal of the PROVIDE system is to allow users to observe and control a program's execution at a suitable level of abstraction [Moher, 1988]. To this end PROVIDE enables users to specify any program objects of interest; graphical views of these objects are then allocated a permanent display area and these views are automatically maintained during program's execution. Another significant

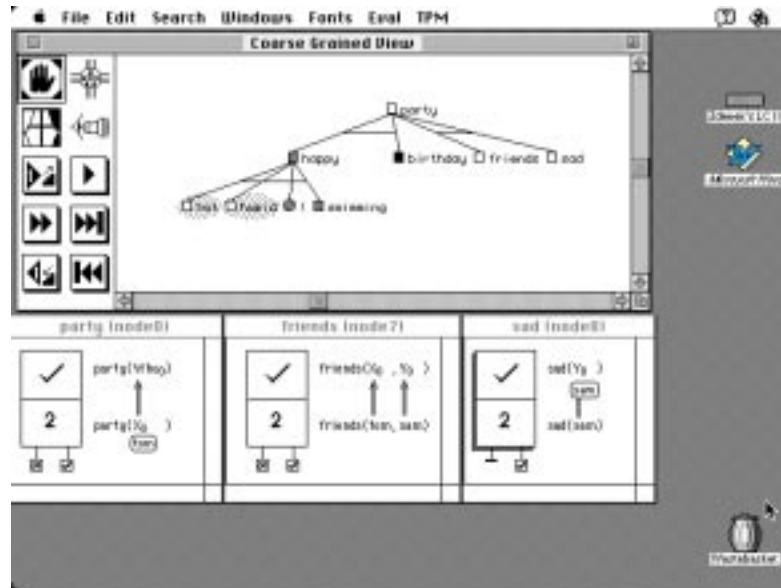


Figure 4.28: A screen shot taken from TPM version 1.11. The coarse-grained view at the top shows the complete execution space of the program, the individual AORTA diagrams at the bottom show a fine-grained view of the program’s individual goals.

feature of PROVIDE is its playback facility in which users can control the apparent speed and direction of execution.

Eisenstadt and Brayshaw - TPM

TPM is a visualization tool for tracing Prolog programs. Like ZSTEP and PROVIDE, TPM supports the bi-directional navigation of a program’s execution. The visualizations in TPM are available as coarse-grained and fine-grained views that share a common “goal tree” metaphor illustrating the structure of the program.

The coarse-grained view represents each goal in the execution of a Prolog program as a node in a graphical tree; squares indicate user-defined goals, circles indicate system primitives, and triangles indicate compressed sections of the tree (see Figure 4.28). The colour of each node indicates the goal’s current state; white nodes (green on a colour display) have been successful, white nodes with a thick outline are currently pending, black nodes (red on a colour display) have failed and grey nodes (pink on a colour display) were initially successful but failed during backtracking. The fine grained view represents the Prolog goals using “AORTA” diagrams i.e. And/OR Trees-Augmented diagrams, which explain the fine-grained details of goal unification.

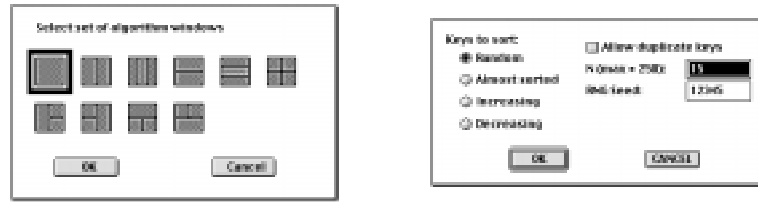


Figure 4.29: A pair of screen shots depicting the set-up phase of a BALSAs session for a number sorting algorithm. The first screen view (left) illustrates the display layout selection dialogue in the centre of the screen. The second screen view (right) illustrates the parameter selection dialogue. In this particular example the user may select the initial organization of the numbers (currently set to a random ordering), the number of numbers to be sorted, and the random number generator's initial seed value.

A more recent extension of this project produced an information management system to support the production of graphical program tracers called “MRE” (the Multiple Representation Environment). MRE has been applied to produce a trace tool for programs written in Parlog, a parallel version of Prolog, see [Brayshaw, 1990] and [Brayshaw, 1994]. A world wide web version of TPM has also recently been implemented in Java for The Open University's Internet Software Visualization Lab (“ISVL”) [Domingue and Mulholland, 1997a], [Domingue and Mulholland, 1997b], see <http://kmi.open.ac.uk/people/paulm/isvl.html>

Further information regarding TPM can found in [Eisenstadt and Brayshaw, 1987], [Eisenstadt and Brayshaw, 1988], or on the world wide web, see <http://kmi.open.ac.uk/kmi-misc/tpm/tpm.html>

Brown - BALSAs

The Brown ALgorithm Simulator and Animator, “BALSAs,” was developed by Marc Brown at Brown University, Rhode Island. BALSAs was the first algorithm animation environment to support a high-level user interface, enabling users to interact with the dynamically changing graphical representations of their programs [Brown and Sedgewick, 1985]. BALSAs was designed as an educational aid to support the teaching of computer algorithms.

Interaction with BALSAs is based around four different user types; the “Algorithm Designer,” the “Animator,” the “Scriptwriter” and finally, the “End User.” The *algorithm designer* provides the programs to be animated, identifies any “interesting events” which need to be visualized, and

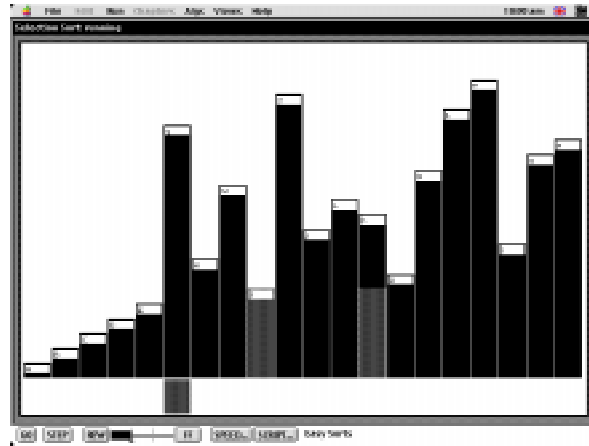


Figure 4.30: A screen shot depicting the run phase of a Balsa session for a number sorting algorithm. The numbers are represented by vertical columns, the magnitude of each number is represented by the height of the corresponding column. As the numbers are sorted by the algorithm, the columns are moved and reordered by height.

contributes to the design of the graphical representations used. The *animator's* task is then to implement the views that make up the graphical presentations. The *scriptwriter* is the person who constructs the scripts for the animation, i.e. what information is shown to the end user and when. Finally, the *end user* makes use of these scripts to view the dynamic graphical representations of a program's algorithm.

The interaction style for the end user is referred to as a “set-up and run” cycle [Brown, 1988]. In the set-up phase the end user arranges the display layout, the algorithms they wish to view, and the parameters they want to associate with each algorithm (including its input generator and output views, see Figure 4.29). Once set up the end user runs the algorithm and observes the results (see Figure 4.30).

BALSA does not support the bi-directional control of the program's execution. The user can either run the program and stop at the next stoppoint, pause at the next stoppoint, stop at the next steppoint, pause at the next steppoint or reset the program back to the start of the execution. The terms “stoppoint” and “steppoint” are taken from Mac Pascal; stoppoints are more commonly known as breakpoints, i.e. points inserted into the program to stop its execution, steppoints are equivalent to the steps of the program's execution i.e. a steppoint occurs after every command.

Further information on BALSA can be found in [Brown and Sedgewick, 1985], [Brown, 1987], or

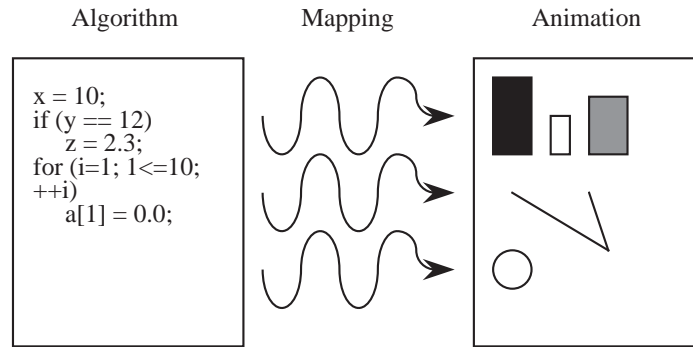


Figure 4.31: John Stasko's algorithm animation framework as used in TANGO. This figure was taken from [Stasko, 1989, page 34].

[Brown, 1988].

Stasko - TANGO

TANGO the Transition based ANimation GeneratiOn framework and system was developed by John Stasko while at Brown University. TANGO was devised for describing, specifying, analyzing and formalizing the elements involved in animating algorithms [Stasko, 1989]. The framework contains three primary components; namely: the “algorithm,” “mapping” and “animation” components (Figure 4.31).

The *algorithm component* adopts an event-driven approach in which any events important to the algorithm's semantics are identified by the algorithm designer and are referred to as “algorithm operators.” These are then used to model procedure calls, mapping the algorithm to the animation. The procedure calls are then used to create the animation control file which constitutes the *mapping component* of the framework. The *animation component* contains the graphical objects, whose location, size and colour will change during animation, and the operations that control the animation. The approach devised here for generating smooth animations is referred to as the “Path Transition Paradigm” [Stasko, 1990].

Four abstract data types are used within the path transition paradigm; “images,” “locations,” “transitions” and “paths.” *Images* are either “primary images” such as lines, rectangles, circles and text, or “composite images” which are collections of primary images with specified geometric relationships. *Locations* are simply positions within the animation co-ordinate system, identified by

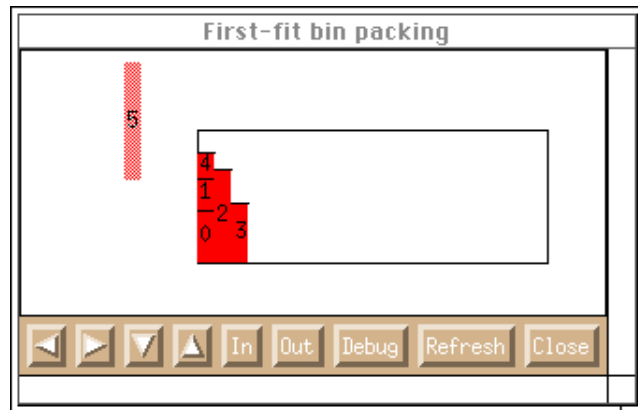


Figure 4.32: A screen view taken from a TANGO animation of a first-fit binpacking algorithm. The elements are inserted into the rectangle and tried against each column position until a large enough free-space is found to house them. The control bar shown at the bottom of the figure allows the user to pan around the view, zoom in and out, switch the debugger on/off, alter the refresh rate, and close the view.

an (x, y) co-ordinate pair. The *path* is an ordered sequence of (x, y) co-ordinate pairs where each pair designates a relative offset from the previous position, and a relative time component used to control the smoothness of the animation. Finally the *transition* component provides the animation with actions to modify the attributes of the image. An example screen view taken from a TANGO animation is given in Figure 4.32.

Further information on TANGO can found in [Stasko, 1989], [Stasko, 1990], or on the world wide web, see

<http://www.cc.gatech.edu/gvu/softviz/SoftViz.html>

Domingue, Eisenstadt and Price - VITAL AND VIZ

The VITAL project was a four and a half year ESPRIT II research and development project, completed in April 1995, involving nine organizations in five different countries. The aim of the project was to provide both methodological and software support for the development of large, industrial, embedded Knowledge-Based System (“KBS”) applications. SV was seen as an opportunity to enhance the users’ control of the individual tools within the VITAL Workbench. In order to support this a separate visualization framework and software library called “VIZ”³ was created [Domingue et al., 1992]. VIZ

³Note the “Viz” visualization framework should not be confused with the “Vis” GA visualization tool developed by Annie Wu (see Section 4.2.1).

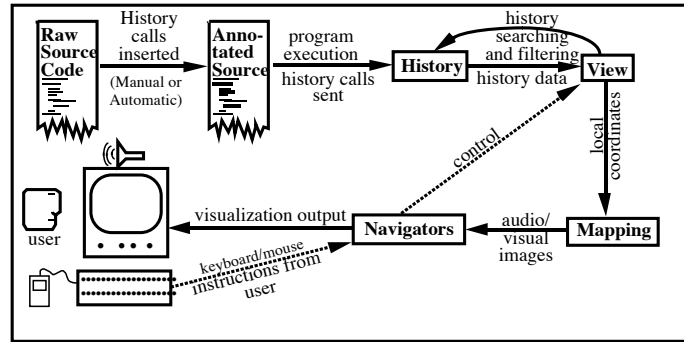


Figure 4.33: The architecture of VIZ. This figure was taken from [Domingue et al., 1993, page 9].

enables the user (i.e. KBS developer) to define and construct visualizations of their systems using a very high level programming language. A program’s execution data is stored in a history database which is used as the basis for creating different views of that program’s execution. These views are then made available to the user who can choose which views are displayed.

To orchestrate this VIZ uses a story-telling metaphor in which the program’s elements (i.e. functions, data structures, lines of code, etc.) are referred to as “players.” The players are identified by the user and annotations are made either to the code or the code interpreter, such that the player’s values are recorded in the History database when interesting “events” occur. A diagram of the VIZ architecture showing the different sub-components of VIZ is given in Figure 4.33.

There are four main components to VIZ, namely the “History,” “Views,” “Mappings,” and “Navigators” components. The *history* component holds a record of all key events that occur over the duration of the program’s execution. The *views* component provides the styles in which a particular set of players, states, or events can be presented. The *mappings* are the encodings used to present the players’ state changes, either graphically, or audibly within each view. Finally, the *navigators* are the tools or techniques used to interact with the user. They allow the user to traverse a view, move between multiple views, change scale, compress or expand objects, and move forwards or backwards through the program’s execution.

The VIZ visualization framework and software library is capable of producing not only program visualizations (i.e. program data and code visualizations) but also algorithm animations. The extent to which the VIZ framework and library is used within the VITAL project is illustrated in Figure 4.34. The Problem Solving Architecture and Code Visualizations are examples of program visualizations,

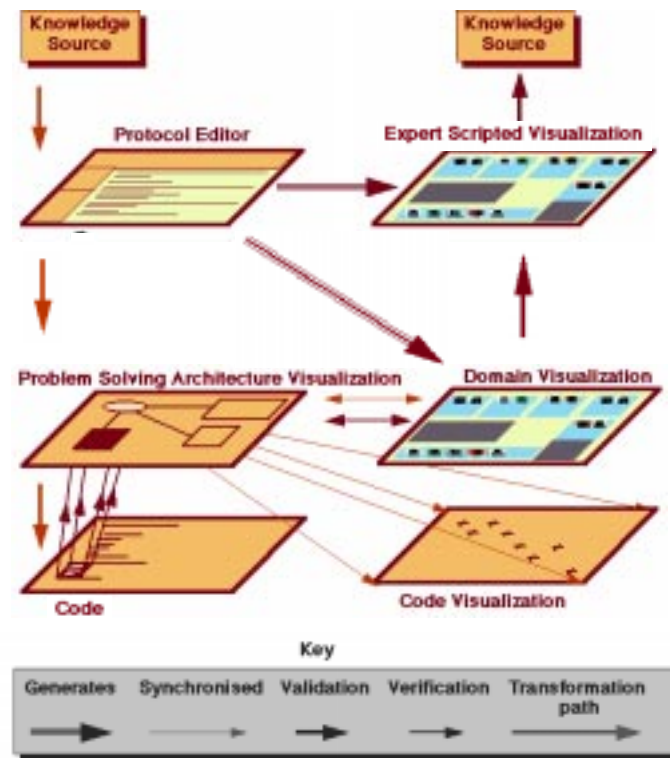


Figure 4.34: An illustration of the software visualization support provided by VITAL. This figure was taken from [Domingue, 1995, page 8].

they closely illustrate the actions of the code and states of the data being manipulated by the KBS. The Domain and Expert Scripted Visualizations are similar to algorithm visualizations where abstract representations are used to illustrate the KBS's operations.

Further information on the VIZ framework and the VITAL Workbench can be found in [Domingue et al., 1992], [Domingue et al., 1993] and on the world wide web, see

<http://kmi.open.ac.uk/people/john/sv/viz/viz.html>

<http://kmi.open.ac.uk/people/john/vital/vital.html>

Brown and Najork - ZEUS

After developing Balsa Marc Brown went to work at the Digital Equipment Corporation (DEC) where, along with Marc Najork, he developed an algorithm animation system called ZEUS. This was designed to provide support for both algorithm animation and multi-view editing.

The use of annotations to indicate “interesting events” in an algorithm is still used, however, added features include the use of objects, strong typing, parallelism and the graphical development of views

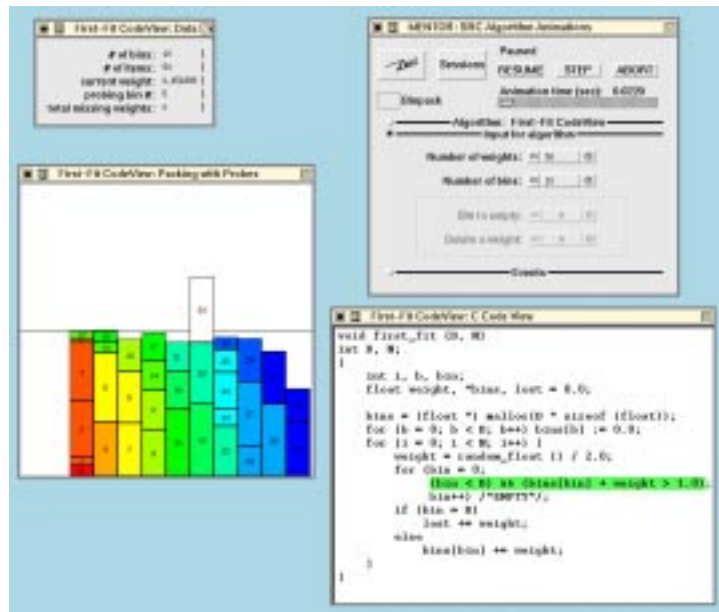


Figure 4.35: A screen shot taken from a ZEUS binpacking algorithm animation. A control panel is shown in the top right window, a code visualization is shown in the bottom right window, an algorithm animation is shown in the bottom left window, and the algorithm's progress is shown in the top left data window.

[Brown, 1991]. The use of objects encourages the reuse of code and facilitates the construction of composite views. The introduction of a graphical editor aids the construction of new view components and the adoption of strong typing provides an opportunity for generating automatic visualizations. A screen shot taken from a ZEUS binpacking animation is given in Figure 4.35.

Further information on ZEUS can be found in [Brown, 1991], [Brown and Hershberger, 1992], [Brown and Najork, 1993], or on the world wide web, see

<http://www.research.digital.com/SRC/zeus/home.html>

Stasko - PARADE AND POLKA

After John Stasko developed TANGO he moved to the Georgia Institute of Technology where he created PARADE, a PARallel program Animation Development Environment. The focus of the PARADE project was to enable the use of “application-specific” visualization to assist the debugging and correctness-checking of parallel programs. Application-specific program views in this context are defined as views that illustrate the program’s semantics, its fundamental methodologies and the inherent application domain.

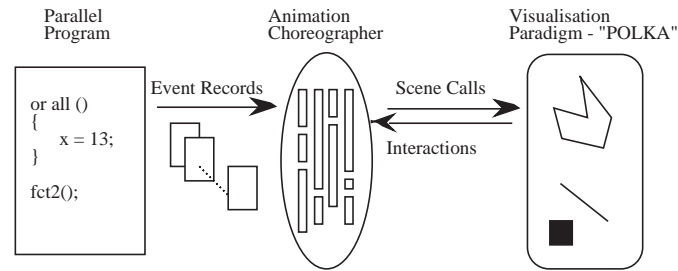


Figure 4.36: An overview of PARADE highlighting its three major components; the “Parallel Program” component extracts the information required for producing the visualizations, the “Animation Choreographer” gathers the program information from the parallel program component and organizes it into a preferred format, and the “Visualization Paradigm” takes the choreographed program details and presents them in an apparently continuous smooth animation to the user. Any user interaction is passed to the animation choreographer by the visualization paradigm where it is acted upon. This figure was taken from [Stasko and Kramer, 1992, page 4].

PARADE is made up of three components; the “parallel program,” “animation choreographer,” and the “visualization paradigm” (Figure 4.36). The *parallel program* component extracts the necessary program information on which to base the views. The *animation choreographer* is responsible for the gathering of the program information and its subsequent organization into a preferred structure identified by the user (via the visualization paradigm). The third component, the *visualization paradigm*, passes the user’s actions back to the animation choreographer and presents the choreographed program details in a smooth animated form. The visualization paradigm in PARADE is called POLKA (Parallel Object-oriented Low Key Animation) [Stasko and Kraemer, 1992]. POLKA is an object-oriented system written in C++ that provides high-level graphical-object primitives and motion primitives for the construction of algorithm visualizations and animations. POLKA is available for both the X Windows and Silicon Graphics GL systems; the Silicon Graphics GL version supports the use of 3D graphics.

The POLKA animation methodology is a combination of principles from the path transition paradigm [Stasko, 1990] and traditional 3D production animation systems. Figure 4.37 illustrates the hierarchy of a POLKA animation. An animation is made up of a series of Views with each view being made up of “Locations,” “Actions” and “AnimObjects.” An *AnimObject* is the base class for all graphical objects (either 2D or 3D); objects are created by the “Originate” method and deleted by the “Delete” method. *Locations* in POLKA can be used to reference and remember important positions

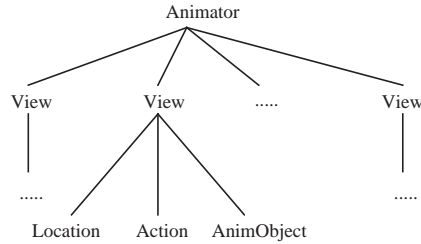


Figure 4.37: A hierarchy diagram illustrating the structure of a POLKA animation. The animator module controls the smooth animation of all the views by ensuring that each animation action is allocated a time-frame. This figure was taken from [Stasko and Kramer, 1992, page 5].

for later use. Locations are real-valued (x, y) markers in the view co-ordinate system. The *Action* class supports the simple movements or changes to be made to the *AnimObjects*, an action object has a type such as “move,” “color,” or “resize” and a list of (x, y) offset pairs defining a two dimensional sequence in the view co-ordinate system. The most significant feature of the POLKA system is its support for concurrent animation that accurately illustrates parallel program concurrency. This is enforced by the programming of each *AnimObject* with actions to occur at particular view frame times. The “Animate” method within the animator class then checks all of the *AnimObjects* for each view and ensures that any actions programmed to occur at the current frame time are executed and the appropriate “update” and “draw” methods are invoked.

POLKA maintains the simple modification of graphical objects along paths approach cultivated in Stasko’s path transition paradigm and adds the capability to program actions into objects at desired animation times. Two screen images illustrating both 2D and 3D visualizations from POLKA are shown in Figure 4.38. The view on the left of the first image is a “blocks view” showing each element in an array as a block whose height indicates the element’s value, and horizontal position indicates its position in the array. The view on the right is a “chart view” in which the horizontal lines are used to represent the swapping of elements; the start and end points of these lines indicate the positions of the elements being swapped. Colour is used in both views to indicate the partitioning of the array.

The second 3D image shows a quicksort algorithm. In this visualization the small blue boxes to the right represent the elements being sorted, the position of each blue box on the y axis indicates the element’s relative value, and its position on the z axes (depth) indicates the elements position in the array. The multicoloured “exchange” planes to the left of the blue boxes illustrate the algorithm’s

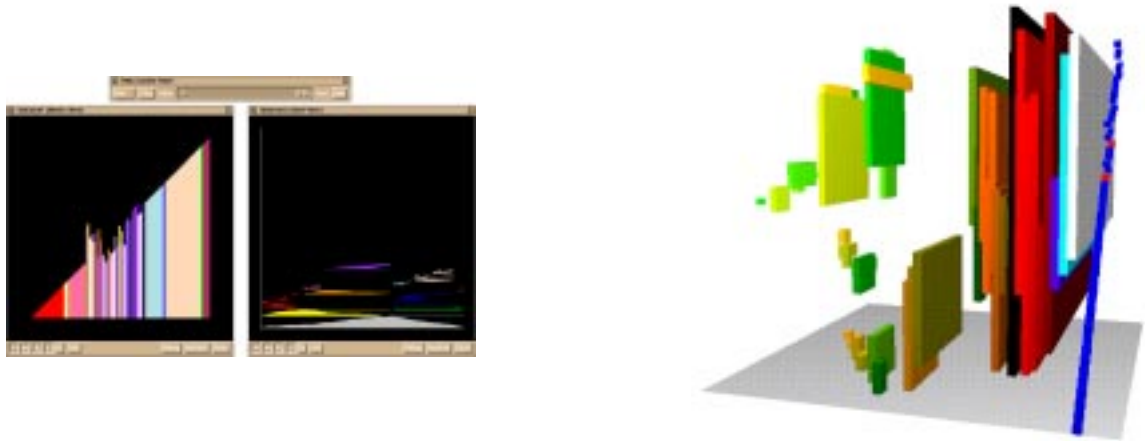


Figure 4.38: Two screen shots showing 2D and 3D POLKA visualizations. The 2D visualization on the left shows the execution of a parallel quicksort algorithm, this visualization contains a control panel (top), a blocks view (left, height = value, horizontal position = position in array), and a chart view (right, vertical position = execution time, horizontal lines = swapping elements). The 3D visualization shown on the right shows the execution of a quicksort algorithm in a single 3D view (y axis = element value, x axis = execution time, z axis depth = position in array).

history from start to finish, shown from right to left.

Further information on PARADE and POLKA can be found in [Stasko, 1995], [Stasko and Kraemer, 1992], [Stasko, 1994], or on the world wide web, see

<http://www.cc.gatech.edu/gvu/softviz/SoftViz.html>

<http://www.cc.gatech.edu/gvu/softviz/parviz/polkaanim.html>

4.2.3 Information Visualization

Shneiderman, Osada and Ahlberg - DYNAMIC QUERIES

“Dynamic Query Interfaces” seek to apply the principles of direct manipulation to database query methods [Shneiderman, 1994]. Shneiderman identified four defining features of dynamic queries:

1. The visual presentation of a query’s components and results.
2. Rapid, incremental and reversible control over a query.
3. Selection by pointing rather than typing.

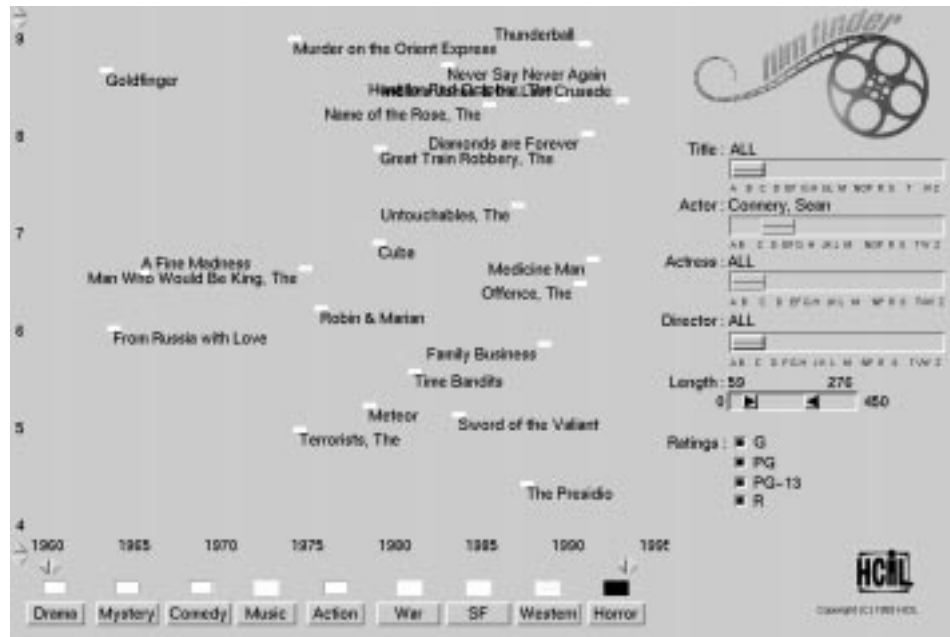


Figure 4.39: The FILMFINDER system which uses alphasliders to identify; film titles, leading actors, leading actresses, directors, and the film length. The x axis is used to indicate the year of release and the y axis indicates its popularity through cinema ticket sales.

4. Immediate and continuous feedback.

An example application which uses the dynamic query approach is the “FILMFINDER” system [Ahlberg and Shneiderman, 1994]. In FILMFINDER a database of film details are accessed through the use of alphasliders and buttons, with the resulting information being displayed in a 2D scatterplot. An example screen shot of the FILMFINDER system being used to find a selection of films starring Sean Connery is given in Figure 4.39. This and other FILMFINDER views are available on the world wide web, see <http://www.cs.chalmers.se/SSKKII/ivee-dumps/filmfinder.html>

An “AlphaSlider” is an example of a dynamic query interface [Osada et al., 1993]. Continuous feedback keeps the user informed of their current position within the data set. A rectangular button is used in a range-defining alphaslider to identify a range of interest. Dragging the left and right hand edges of the rectangular button defines the start and end of the data range, and the rectangle itself can also be dragged to pan across the data set.

A selection of some of the work done using dynamic queries can be found in Christopher Ahlberg’s world wide web site on information visualization and exploration, see

<http://www.cs.chalmers.se/SSKKII/ivee.html>

More recently a project exploring the use of dynamic queries for SV has started at Washington State University; further information can be found at the exploratory visualization world wide web site, see

<http://swarm.cs.wustl.edu/~roman/QueryVis.html>

4.3 Summary of the Contributions Made

In this chapter the existing visualization support tools and techniques suitable for displaying the key characteristics of GAs have been introduced, and their suitability for supporting the user's understanding of the GA's search behaviour has been discussed. The final summary draws together the contributions made and remaining work to be done.

The conclusions of the user study highlighted a need to support the user's understanding of the GA's search behaviour. Of the key characteristics discussed, visualizing the GA's sampling of the search space is most effective for illustrating the GA's search behaviour. Although measures of the populations' diversity or problem complexity may be useful to indicate the GA's search behaviour, actually seeing the GA's search behaviour gives the user a more direct insight. The only problem with this approach is representing the high dimensional search space on a two dimensional screen.

Other key characteristics of significance for this project are the navigation of the GA's execution and visualizing the quality of the GA's solutions. The provision of bi-directional navigation support for viewing the GA's execution generation by generation and the potential use of dynamic queries for exploring sections of the search space, are two new approaches for GA navigation which have proven to be extremely useful within the respective fields of SV and information visualization. Visualizing the quality of the GA's solutions using a fitness versus time graph is the most common form of GA visualization simply because it shows the GA user something that they need to know. The provision of a fitness versus time graph is an essential view that can be augmented either with a vertical line to highlight the current generation, or with a rectangle to highlight the range of generations and fitness ratings being displayed in other views.

Editing the GA's parameters and operators may be useful for the GA user yet it is not directly a

part of the GA's visualization, however, if visualization support for understanding the GA's search behaviour is not available then it will be difficult for the user to judge the effects of any changes made except those which directly influence the quality of the final result. Editing the chromosomes in the GA's population is not a common step involved in using a GA, but it may be a useful way of introducing problem specific knowledge or exploring the GA's behaviour. Although human-intervention of this nature could be considered intrusive or even damaging to the GA's operation, such arguments are outside the scope of this project, where the primary concern is for supporting the GA user. Supporting the editing of the GA's chromosomes may be achieved by providing an interactive search space visualization. This could be used to explore unconsidered sections of the search space independently of the GA's population, seeding the population with specific chromosomes, or (if the user wishes) for moving chromosomes in the population to new positions in the search space.

Visualizing the chromosomes' (genotypes or phenotypes) involves the use of detailed (problem-independent or problem-dependent) views of the solutions being considered by the GA. Viewing all the chromosomes in a population produces a lot of information and unless the user is specifically interested in examining the population's chromosomes (as they are with VIS in the Virtual Virus project), such visualizations should be used selectively so that the GA user can identify the individuals that they are interested in. The last key characteristic examined was visualizing the GA's operators, which is an effective educational visualization but is not an informative visualization of the GA's overall search behaviour.

In conclusion, visualizing the GA's sampling of the search space, navigating the GA's execution and coverage of the search space, and displaying the quality of the solutions found by the GA, are the three most important forms of visualization support for the user wanting to understand the GA's search behaviour, the provision of which is discussed in the next chapter.

Chapter 5

Visualization Design Rationale

The aim of this project is to investigate how SV technology can be applied effectively to support peoples' understanding of EC. As noted in Section 1.3, SV is intended to facilitate the human understanding and effective use of computer software through the use of crafts such as typography, graphic design, animation and cinematography, with modern human-computer interaction technology. An appreciation of the generic guidelines available in the Human-Computer Interaction ("HCI") literature is important for SV design, as even the best graphical representation will be ineffective if presented through a weak interface. HCI sources of reference include [Preece et al., 1990], [Thimbleby, 1990], [Preece et al., 1994], and [Shneiderman, 1998]. Some of the seminal papers in this area have been collated and republished in [Norman and Draper, 1986] and [Baecker and Buxton, 1987].

Within HCI a number of Psychologists have been exploring the use of graphical representations and their role in the context of problem solving activities, such as computer programming. For examples see: [Green, 1982], [Larkin and Simon, 1987], [Baecker and Buxton, 1987, Chapter 7], [Larkin, 1989], [Davies, 1991], [Cox and Bruna, 1995], [Scaife and Rodgers, 1996], and [Petre et al., 1998]. Although an appreciation of these guidelines is important, the design of an SV (as the above definition implies) is formed in the associated craft. For example; when an SV designer is producing a text-based program editor which provides typographic support, the design is rooted in typography. In the case of designing graphical representations of a GAs search behaviour, the visualization is rooted in the craft of graphic design and the psychology of human perception (see, [Bertin, 1981], [Bertin, 1983], [Tufte, 1983], [Tufte, 1990]).

This chapter examines how effective visualization support may be provided for GA users. Section 5.1 explores how visualization support should be conveyed to GA users by comparing the advantages and disadvantages of visualization development frameworks and visualization systems. A visualization design approach based on the principles of graphic design is presented in Section 5.2 and Section 5.3 describes how this approach can be applied to produce graphical representations suitable for illustrating a GA's search behaviour.

5.1 A Framework Approach Versus a Systems Approach

This section explores the differences between frameworks and systems. A system is defined as a complex whole, a set of connected things or parts [Hawkins and Allen, 1991]. In the case of a visualization system this refers to a set of connected visualizations designed to support peoples' understanding of a particular topic. An example of such a system would be the GIGA system described in subsection 4.2.1, [Dabs and Schoof, 1995]. GIGA supports users' understanding of GAs by providing a set of visualizations showing details of the best chromosomes from each generation, the population's convergence, the phenotype of the best individual found so far, and the internal operations of the genetic operators. A framework on the other hand is an essential supporting structure or a basic system [Hawkins and Allen, 1991]. In the case of a visualization framework this refers to a supporting structure that provides the basics for visualization and the opportunity to build and develop further visualizations. An example of this would be the VIZ framework described in subsection 4.2.2, [Domingue et al., 1992].

To summarize this distinction, a system can be thought of as being made up of a set of parts which may be deployed as required, whereas a framework provides a basic set of parts which may be used to develop and build further parts, and is therefore capable of producing systems. Although a system may be extended the point of distinction is that a framework supplies a supportive structure for the future development of new parts whereas a system does not.

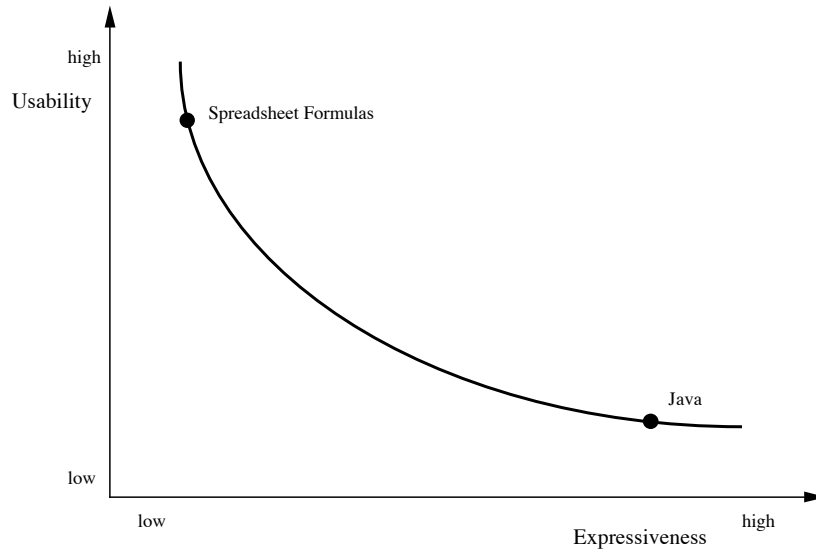


Figure 5.1: The usability-expressiveness curve illustrating the trade off between usability and expressiveness as described in [Repenning and Ambach, 1996].

5.1.1 The Usability-Expressiveness Trade Off

In order to be effective for a wide range of applications a programming environment must be both *usable* and *expressive* [Bell et al., 1991], [Repenning and Ambach, 1996]. . The effort involved in using any environment should be kept to a minimum whilst still enabling the user to realize all of their requirements [Repenning and Ambach, 1996] identified a trade off between these two desirable features along which most programming environments can be located (see Figure 5.1). For example, text based programming languages such as Java or C are more *expressive* than spreadsheet systems such as Excel or Quatro Pro but spreadsheets are more *usable* for accounting tasks than text-based programming languages.

It is proposed that the usability-expressiveness trade off is an inevitable result of introducing the need for human understanding, in that the human perception system can be thought of as having a limited number of input channels (i.e. our five physical senses) each having a limited “bandwidth.” One solution to this has been to develop languages with which a large amount of information can be carried by a relatively few number of signs; examples include mathematics, graphics and spoken languages. Information is encoded in these languages and communicated to other people. The expressive power of the language is inversely linked to its usability, the more expressive the language

the more difficult it is to use because it requires more effort to recall and manipulate the syntax and semantics used. In their approach to designing end-user programming languages Repenning and Ioannidou propose the provision of a layered programming environment [Repenning and Ioannidou, 1997]. The expressiveness of the language decreases with each layer while the usability increases such that the user can choose a layer to work at that suits their current task, providing a sufficient level of expressiveness at a maximum level of usability.

5.1.2 Systems versus Frameworks

Having identified the inherent trade off between usability and expressiveness, where should we place the visualization support needed by GA designers? GAmeter and GIGA, as described in Section 4.2.1, both use a WIMP (“Windows Icons Menus and Pointing”) interface to provide visualization support and although this contributes strongly toward usability for novices, the level of expressiveness in terms of views and view configurations is limited in both cases. The intended user’s profile i.e. their usability and expressiveness requirements, should obviously be taken into account when making such decisions.

As seen in the GA user study, GA users work on a variety of domains and examine a variety of features from their algorithms, and there is no single visualization which best supports all the tasks involved in using GAs. In fact, it is argued here that there is no single set of visualizations that will suit all GA users or that such a set could ever exist. EC is a rapidly developing area of research, feeding on and contributing to parallel research being carried out in a range of fields including evolutionary biology, problem solving and adaptive systems, as well as from other areas within artificial intelligence. As long as this research continues the definition of such a set of *sufficient* visualizations will be impossible as the definition of EC and the visualization requirements of EC designers will continue to change. Therefore, in order for SV to be of practical use to the EC community the importance of expressiveness cannot be overstated. Whilst maintaining a regard for the usability of the visualization environment the users should be in the position to design and develop their own visualizations.

It was for these reasons that a framework approach was adopted in this project. Rather than developing a visualization system incorporating what are currently considered useful visualizations, a framework approach was chosen with which visualizations could be provided either in an off-the-shelf

ready-to-use format, customizable by their users, or designed and developed as required using existing building blocks where available. In this way the users should be able to express what they need (i.e. have *sufficient* expressiveness) using the highest available level of abstraction (i.e. with *maximum* usability).

5.1.3 The Advantages of the Framework Approach

By providing the users with the ability to build up their own graphical representations from the most basic image elements (i.e. points, lines and areas) the user has the opportunity to create their own views. This provides the user with an unlimited freedom of expression but at a cost to usability; in order to develop visualizations in this way (i.e. with points, lines and areas) the user must invest a considerable amount of programming effort. However, the framework approach involves much more than the basic struts and links for building visualizations. Within the framework itself there is a supporting structure, a scaffolding used to support the layered development of visualizations at higher levels of abstraction and with which the users can build their own task specific supporting structures.

By forming a taxonomy of views using an object-oriented inheritance hierarchy, as used in VIZ [Domingue et al., 1992], the basic features of each view can be defined and reused by passing them down the hierarchy to more and more specific views. Furthermore, the views provided in the view hierarchy can introduce good design principles that will be re-used by subsequent view designs, providing at least a good start for the efficacy of the visualizations produced. By adopting the framework approach the user can either use a view directly from the view hierarchy, or introduce a new view by creating a specialist form of the most applicable view and adding their own task-specific features.

Applying a view to produce a visualization requires the provision of a mapping between a chosen view and an item of interest. Linking attributes in the software to image components in this way is a common trait in a variety of SV environments such as TANGO, VIZ and PARADE (see Chapter 4). An adapted version of the VIZ architecture is used in this thesis to produce a GA specific visualization framework called “HENSON.” The design and application of HENSON is explained further in Chapter 6.

Table 5.1: A taxonomy of sign-systems, each system is identified by the type of meaning attributed to the signs, and the system of perception used. This table was taken from [Bertin, 1983, page 2].

MEANING ATTRIBUTED TO SIGNS	FORM OF PERCEPTION	
	EAR - SOUND	EYE - SIGHT
<i>monosemic</i>	Mathematics	Graphics
<i>polysemic</i>	Language	Figurative Imagery
<i>panasemic</i>	Music	Abstract Imagery

5.2 A Principled Approach to SV Design

Visualization is frequently used in science, economics, and statistics as a means for facilitating understanding. For example, visualization is seen as one of the most important techniques used in the field of Knowledge Discovery from Databases (“KDD”):

“The appropriate display of data points and their relationships can give the analyst insight that is virtually impossible to get from looking at tables of output or simple summary statistics. In fact, for some tasks, appropriate visualization is the only thing needed to solve a problem or confirm a hypothesis.” [Brachman and Anand, 1996, page 45].

The point that must be emphasized here is the use of *appropriate* visualizations. In order to produce appropriate visualizations a principled design approach must be taken. Jacques Bertin’s “Semiology of Graphics” identifies the properties of the graphic sign system and a set of rules for applying the graphic system [Bertin, 1983]. Given a specific set of information the task of the graphic designer is to analyse the information identifying the items of interest and through the application of the rules of the graphic system, produce a design whose properties match the properties of the information of interest. This section presents an overview of Jacques Bertin’s design principles and explains how they may be applied to SV design. A further understanding of the principles of graphic design can be gained from the literature, see [Bertin, 1981], [Bertin, 1983], [Tufte, 1983], and [Tufte, 1990].

5.2.1 Analysis of the Information

Graphics is an example of one of our basic sign-systems used for communicating thought (see Table 5.1). Graphics involves the transcription, into the graphic sign-system, of “information” known through the intermediary of any given sign-system. Transcription leads necessarily to the separation of *content* from *form*. Either for studying the means, properties and limits of the graphic system, or planning a design, it is first necessary to separate the content (i.e. the information to be transmitted) from the container (i.e. the properties of the graphic system). Bertin identified the translatable content of a thought as “information,” which is made up of one or more pertinent correspondences between a finite set of variational concepts (i.e. “components”) and an invariant concept (i.e. “invariant”). For example:

“On July 8, 1964, stock X on the Paris exchange is quoted at 128 francs; on July 9, it is quoted at 135 francs.” [Bertin, 1983, page 5].

Here the two components are the number of francs and time, and the invariant is stock X. The invariant and components are used to identify the title of the graphic. Once the invariant and components have been determined the next step in the analysis is to identify the parts of each component, these are called the “elements.” The number of elements in each component is referred to as the component’s “length.” The complexity of a figure is linked to the length of its components.

The components of the graphic sign-system are called the “visual variables.” In order to illustrate each component of information, Bertin identifies three levels of organization upon which each component and visual variable can be located; the “qualitative level,” the “ordered level” and the “quantitative level.” The *qualitative level* includes all the concepts of simple differentiation and involves two perceptual approaches; “is this similar to that?” (i.e. association), and “is this different to that?” (i.e. selection). The *ordered level* involves all the concepts that permit a ranking of the elements in a universally acknowledged manner e.g. “this is more than that and less than the other.” Finally, the *quantitative level* is attained when a countable unit is used e.g. “this is a quarter of that and four times the other.” These levels overlap; what is quantitative is also ordered and qualitative, what is ordered is also qualitative, and what is qualitative is neither ordered or quantitative. It is important that each component be transcribed by a variable having at least a corresponding length

Table 5.2: The levels of organization for visual variables. This table identifies the appropriate level of organization for each of Bertin's eight visual variables. These visual variables support either the perception of associations or dissociations, selective perception, the perception of order, and/or the perception of quantities. This table was taken from [Bertin, 1983, page 69].

VISUAL VARIABLE	ASSOCIATIVE	SELECTIVE	ORDERED	QUANTITATIVE
PLANAR DIMENSIONS	≡	≠	○	∅
SIZE	≠	≠	○	∅
VALUE	≠	≠	○	
TEXTURE	≡	≠	○	
COLOUR	≡	≠		
ORIENTATION	≡	≠ (P & L)		
SHAPE	≡			

and level. Graphics is concerned with the representation of these three levels of organization.

5.2.2 The Properties of the Graphic System

Bertin identified eight visual variables: a visible mark expressing a pertinent correspondence can vary in relation to the two dimensions of the plane, as well as in size, value (i.e. the colour lightness value as defined by a colour's hue, saturation and value tuple), texture, colour (i.e. hue), orientation and shape. Within the plane the mark can represent either a point (a position without area), a line (a linear position without area), or an area. The type of mark used, i.e. point, line, or area, is called the "implantation" or "class" of representation. The use of the two dimensions of the plane is referred to as "imposition," the six remaining visual variables are called the "retinal variables" and their utilization is referred to as "elevation." The level of organization of the plane is maximum and therefore any component of information, whatever its level of organization, can be represented by imposition. However, none of the retinal variables has the capability of the plane to represent any component, and therefore, the level of organization, properties of length, and applicability of the retinal variables, must be taken into account when designing a graphic.

The levels of organization of the retinal variables, like the levels of organization of information,

are either qualitative (involving associative or selective perception), ordered, or quantitative. The levels of organization for each visual variable are given in Table 5.2, where the visual variables are either associative (\equiv), dissociative ($\not\equiv$), selective (\neq), ordered (\circ), or quantitative (\propto).

Associative perception (\equiv) is useful for equalizing a variation and grouping correspondences of all categories with this variation. If the eye can immediately reconstruct the uniformity of a series of undifferentiated points forming a uniform area, in spite of a given visual variation, this variation is associative, otherwise it is dissociative ($\not\equiv$). The reason for this perceptual association is the sign's "weight" or "visibility;" if the visibility of the signs used are equal then they can be associative; if their visibility changes some will appear more powerful than others and are therefore dissociative. Shape, orientation, colour, texture and the planar dimensions are associative, whilst value and size are dissociative.

Selective perception (\neq) is used for identifying the location of a given category, the eye must be able to isolate all the elements of this category and disregard all the others. Providing the perception of any given category is immediate then the sign can be said to be selective. Shape is not selective for either point, line or area implantations and orientation is not selective when represented by area. The remaining visual variables enable selective perception.

Ordered perception (\circ) must be used when comparing two or more orders. When a variable is ordered, it is not necessary to consult the legend to be able to order the categories - the order of the signs is universal and immediately perceptible. Texture, value, size and the two planar dimensions, are ordered; shape, orientation and colour are not.

Quantitative perception (\propto) is used to define a numerical ratio, a simple test for quantitative perception is to ask an observer what the value of the largest sign would be if the smallest sign's value was one. The observer should be able to perceive the numerical ratio between the signs without consulting the legend. Size is the only quantitative retinal variable.

Figure 5.2 gives a detailed illustration of the properties of the eight visual variables; the two planar dimensions, size, value, texture, colour, orientation and shape. The two planar dimensions are the only two variables that can be used to illustrate similarities, differences, orderings and proportions. Similarities can be shown using texture, colour, orientation and shape. Size value, texture, colour and (point and line) orientation can be used to show differences. Orderings can be shown by size,

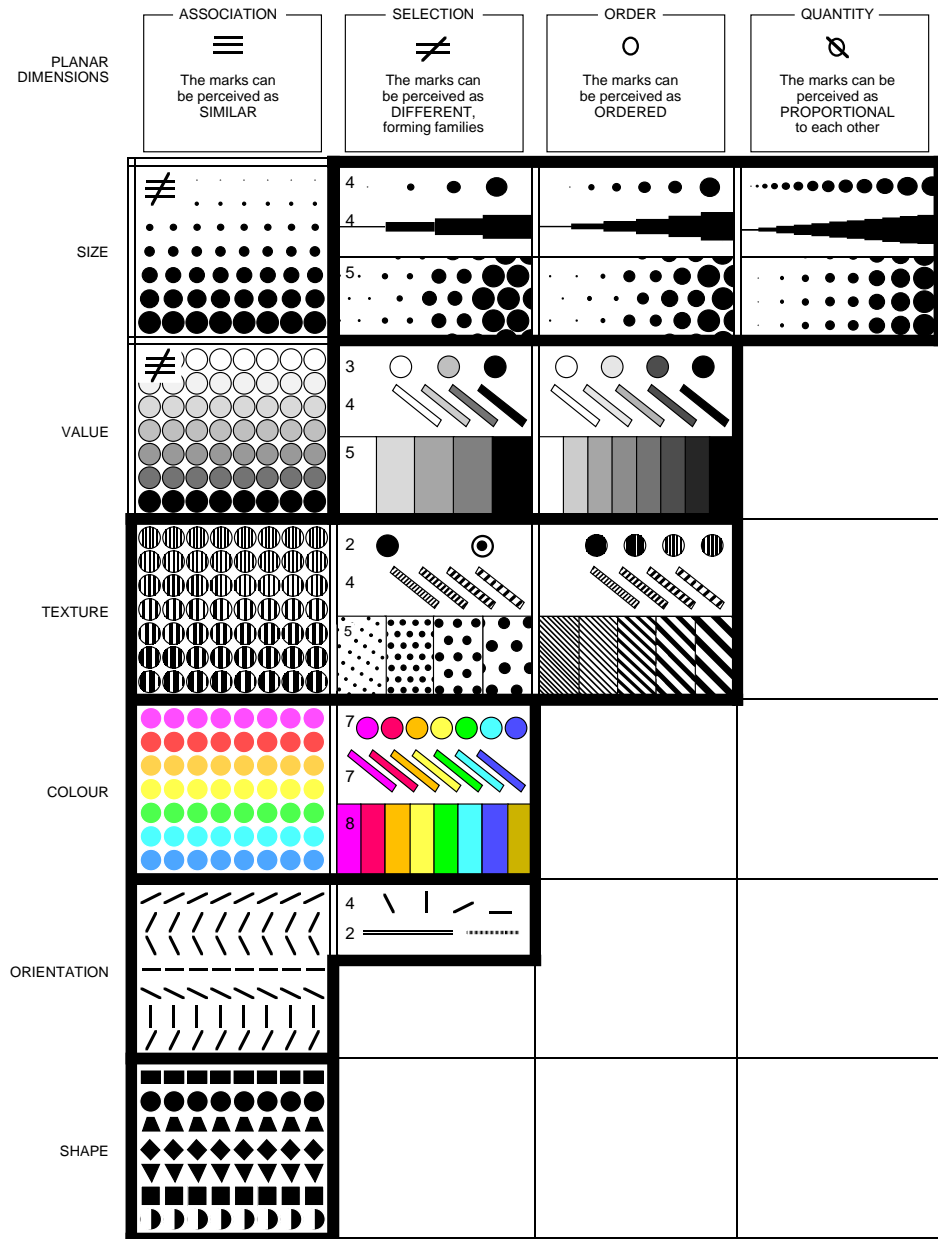


Figure 5.2: An illustration of the levels of organization of each of Bertin's eight visual variables; the two planar dimensions, size, value, texture, colour, orientation and shape, as identified in Table 5.2. Included are examples for point, line and area implantations, the numbers shown on the left of the Selection column indicate the recommended number of levels to support selective perception by each implantation (point, line and area, respectively). This figure was taken from [Bertin, 1983, page 96].

value and texture. Finally proportions can be shown by variations in size.

By identifying the levels of organization in the information to be represented and selecting a set of visual variables with at least the same level of perception the basic form of the graphic can be identified. The design of the graphic should then follow on from the properties of the information and the properties of the visual variables being used. Further details regarding the level of organization, properties of length and applicability of the retinal variables can be found in [Bertin, 1983, pages 69 to 97].

5.2.3 The Rules of the Graphic System

In the course of describing the rules of the graphic system, Bertin discusses image efficiency, image theory, the function of graphics, the construction of graphics and the legibility of graphics.

Image efficiency is linked to the time taken for an observer to obtain the correct and complete answer to a given question. If an observer can answer a question more quickly using graphic A than graphic B, then graphic A is said to be more efficient than graphic B. *Image theory* enables the designer to choose the variables which will construct the most efficient image. This can be summarized in five basic rules:

1. Stages in the reading process. There are three successive operations associated with the reading process: “External identification,” “internal identification,” and the “perception of pertinent correspondences.” *External identification* identifies the components involved, *internal identification* identifies how those components are expressed (i.e. the visual variables used), and the *perception of pertinent correspondences* identifies the answer to a given question.
2. Possible questions - levels of reading. Any question can be defined by its type and level. There are as many types of question as there are components in the information and for each type of question there are three levels; the “elementary level,” the “intermediate level” and the “overall level.” An *elementary level* question will ask about a single element of a component, e.g. “On a given date what is the price of stock X?” *Intermediate level* questions will ask about a group of elements within a component, e.g. “In the first three days what was the trend of the price?” Thirdly, *overall level* questions will ask about the whole component, e.g. “During the entire

period what was the trend of the price?”

3. Definition of an image. An image can be defined as “a meaningful visual form perceptible in the minimum instant of vision.” The most efficient constructions therefore will be those in which any question, irrespective of type and level, can be answered in a single instant of perception, i.e. from a single image.
4. Construction of an image. An image is built upon three homogeneous and ordered variables: the two dimensions of the plane and a retinal variable. All information with three components or less can be shown in a single image. In any construction made for more than three components, certain types and levels of question will necessitate the perception of several images in succession. These will be less efficient than a representation involving a single image and will incur a higher mental cost.
5. The limits of an image. The visual efficiency of a graphic is inversely proportional to the number of images necessary for the perception of the data. Therefore, the designer should identify a set of preferred questions and construct the graphic such that these questions can be answered in a single instant of perception from a single image, and the less useful, or less likely questions to be asked, can be answered after several instants of perception, i.e. from several images.

Bertin’s exploration of the *function of graphics* identifies three functions: “recording information,” “communicating information,” and “processing information.” A graphic used for *recording information* creates a storage mechanism that saves the observer the effort of memorization. An example would be an ordnance survey map which is commonly used as a source of reference for recording geographical information. In this case the graphic must be comprehensive but not necessarily memorizable. Graphics with the function of recording information are referred to as “inventories” which favour reading at the elementary level. Inventory graphics can be complex figurations, with multiple images, limited only by the rules of legibility.

Graphics for *communicating information* on the other hand must be memorizable but not necessarily comprehensive, as they are required to inscribe the information in the observers mind. An example would be a weather map which presents a simplified image of a geographical area which can be easily recalled. Graphics for communicating information must be simplified drawings capable of

being memorized, which are also referred to as “messages.”

The third function of graphics identified by Bertin was for *processing information*, which should be both memorizable and comprehensive, giving a justified simplification of the information. They must not eliminate any part of the information, but enable “processing” such that by using the order and classification to discover groupings within the information new components or categories can be formed which may be easier to memorize. An example is of an underground train map; although the map shows the entire underground train system it can be processed in such a way as to identify and recall individual routes of interest.

Although information with three components or less represented in a single image can fulfill all three functions, information involving more than three components must be constructed differently according to the nature of the preferred questions and hence the intended function of the graphic. In order to produce the most efficient graphic for any given set of information, Bertin proposed three general rules regarding the *construction of graphics*:

1. “To represent the information in a single image, or in the minimum number of images necessary (to render it perceptible in its entirety, in the minimum number of instants of perception), is the first rule of graphic construction.”
2. “To simplify the image without reducing the number of correspondences is the general rule which applies to any information having one or several reorderable components.” This is known as the diagonalization of diagrams or transformation of networks
3. “To simplify the image by reduction and thus create a clear and efficient message is the general rule which applies to any information having several ordered components.” This involves the elimination of details, referred to as the “smoothing of curves” in diagrams, and “regionalization” and “generalization” in maps

Simplifying a diagram containing one or several reorderable components is achieved by a process known as “diagonalization” which simply reorders the components such that they form diagonal trends. A similar approach is used to simplify re-orderable components in networks, in which case it is referred to as “transformation.” The network nodes are re-ordered to minimize the length of the paths linking nodes and the number of crossing paths.

Figure 5.3: An illustration of Bertin's fifteen standard schema for constructing diagrams, networks and maps. This figure was taken from [Bertin, 1983, page 172].

The simplification of ordered components requires the reduction of unnecessary detail from the image; for diagrams this is referred to as the “smoothing of curves,” and for maps this is known as “regionalization” and “generalization.” Examples include the use of trend lines in graphs, and simplified coastlines or country borders in maps.

In order to support the application of these three rules, Bertin produced a set of standard design schema for diagrams, networks and maps (see Figure 5.3). These embody Bertin’s rules of graphic construction; the connected horizontal and vertical arrow lines shown in Figure 5.3 indicate the use of the two planar dimensions. Unconnected horizontal and vertical arrow lines indicate the use of multiple (i.e. “ $\times n$ ”) images. The clockwise circular arrow lines in the network schema indicate the location of network nodes. The short diagonal arrow lines indicate the use of one of the six retinal variables to illustrate a component of information, and finally, the use of an anticlockwise arrow line indicates that no retinal variables are used in the graphic. The resulting function of each graphical schema is also indicated below each section of the figure.

Finally, Bertin’s rules regarding *legibility* focus upon the “graphic density,” “angular separation” and “retinal separation” of graphics. Bertin proposed that there is an optimum number of marks per square centimeter in any given figure. Although the optimum number was considered to vary with the number of different marks being used, the implantations being applied, the retinal variables

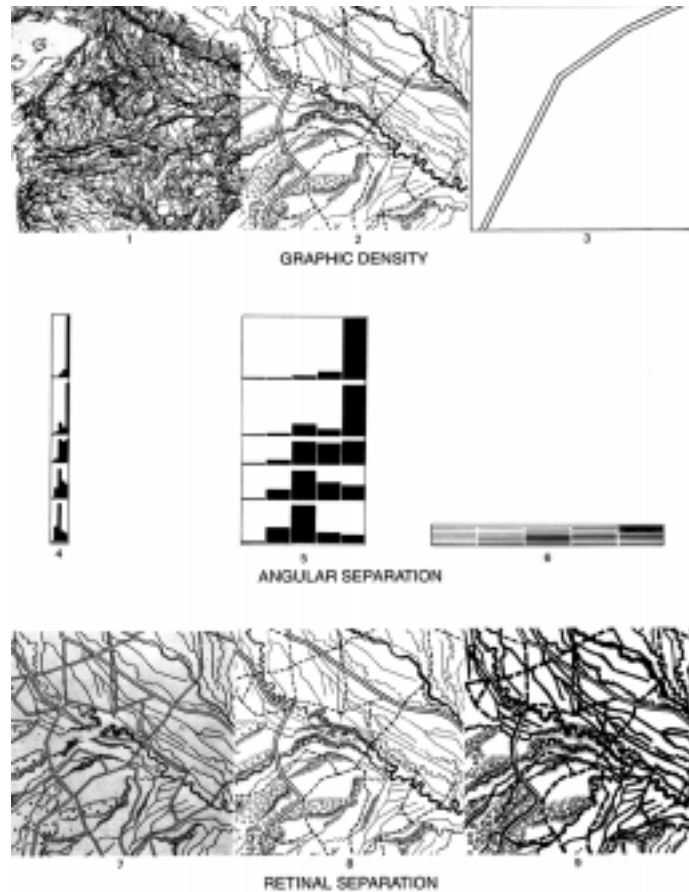


Figure 5.4: An illustration of Bertin's general rules of legibility, regarding graphic density (parts 1, 2, and 3), angular separation (4, 5, and 6), and retinal separation (parts 7,8, and 9). This figure was taken from [Bertin, 1983, page 174].

being employed and the observer's reading habits, a maximum graphic density of 10 signs per cm^2 was recommended. Similarly, in order to support the legibility of an image, the full range of perceptible differentiation afforded by the visual variables should be such that the use of the two planar dimensions avoids "squashing" the image and limiting the angular differentiation of each sign, and the retinal variables are used to emphasize the differences between the meaningful and meaningless marks, separating the "figure" from the "ground" and the "form" from the "content." Figure 5.4 illustrates the correct and incorrect application of these rules.

5.2.4 Design Summary

The above five rules of the graphic system, regarding image efficiency, image theory, the function of graphics, the construction of graphics, and the legibility of graphics, were put forward by Bertin

as a set of guidelines for graphic design. These guidelines are recommended as a generic SV design approach and are adopted in this project to design GA visualization support, as described in the following section.

5.3 The Principled Design of Search Space Visualizations

This section investigates how the operation of a GA can be represented to the user at an appropriate level of abstraction. As noted in the previous section the first stage in producing any form of graphical representation is the analysis of the information to be shown, i.e. identifying the set of variational concepts (i.e. “components”) and an invariant concept (i.e. “invariant”). In the case of SV identifying the information to be shown is often necessary before the design process can begin. Section 3.3 identified a set of typical questions that some GA users appeared to be asking themselves when exploring the operation of their algorithms. This section identifies and analyses the information required to answer the users’ queries (Subsection 5.3.1), presents a set of graphical schemata suitable for displaying this information (Subsection 5.3.2), and proposes a number of visualization designs appropriate for supporting the users’ understanding of their GA’s search behaviour (Subsection 5.3.3). This section finishes with a discussion (Subsection 5.3.4) and concluding summary (Subsection 5.3.5).

5.3.1 Identifying Relevant Information

The responses given in the GA user study identifying the users’ opinions regarding GA visualization (i.e. Questions 7, 8 and 9) repeatedly make reference to the GA’s search behaviour, specifically with regard to the convergence and diversity of the population, identifying clusters, niches, similarities and the population’s stability (see Section B.3). Furthermore, when asked directly about visualizing the rate of change in the populations’ fitness ratings (Question 7.3) respondents from all three user groups (theory, research and applications) noted that visualizations based solely on illustrating fitness were often insufficient. From these responses it was concluded that in practice GA users are interested in seeing how their algorithms search the problem space, and not just in the results that their algorithms achieve. From the findings of the user study the following set of users’ questions were derived:

- How diverse/converged are the chromosomes in the population?

- Are there clusters of chromosomes forming during the GA's run?
- How does the local structure of the chromosomes affect the chromosomes' fitness ratings?

Each of the above questions relate to the chromosomes' diversity, their location in the search space, and the correlation between the chromosomes' local structure and their fitness ratings. In order to answer these questions, visualizations must be produced that can illustrate the following information;

- the values of the chromosomes in the population,
- the chromosomes' position in the search space, and
- the relationship between the chromosomes' fitness ratings and local structure.

Hence, the visualizations must show the search space, the chromosomes' positions in the search space and the chromosomes' fitness ratings. The level of organization for the chromosomes' position in the search space is essentially ordered by the chromosomes' similarity to one another, such that genetically similar chromosomes are placed near each other and genetically different chromosomes are placed apart. The organization of the fitness ratings, although generally measured as quantities, should actually be organized by quality, such that the user can form associations between similarly fit regions of the search space and selectively identify differences in the fitness ratings between different regions of the search space, in order to derive the relationship between the chromosomes' local structure and their fitness ratings.

Therefore, visualizations of the GA's chromosomes in the search space are essentially made up of one or two ordered components representing the GA's chromosomes in the search space (using either one or two (planar) dimensions) and one component illustrating the chromosomes' fitness ratings (using either a quantitative, ordered or qualitative visual variable).

5.3.2 A Set of Applicable Graphical Schemata

Figure 5.5 shows two graphical schemata for showing a GA's search space. These can be applied to produce 2D or 3D visualizations of a GA's fitness landscape, examples of which were presented

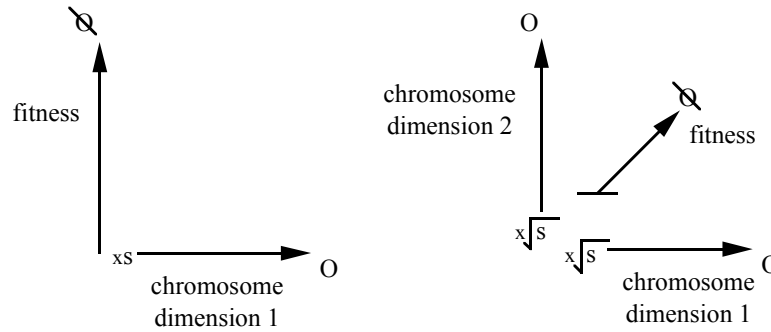


Figure 5.5: The most efficient graphical schemata applicable for producing 2D and 3D visualizations of a GA's fitness landscape.

in Section 4.1.5. The examples of two dimensional fitness landscapes show the chromosomes' fitness ratings on the y axis and their positions in the search space on the x axis, see [Collins, 1993] and [Spears, 1994]. Examples of three dimensional fitness landscapes generally use the two planar dimensions to show a perspective projection of the chromosomes' fitness ratings on the y axis and their positions in a two dimensional search space on the x and z axes, see [Spears, 1994] and [Nassersharif et al., 1994].

The second graphical schema described in Figure 5.5 identifies three components, two showing the search space and one for showing the fitness rating. This design produces an image more compact than the 2D fitness landscape, in that both planar dimensions are used to show the search space rather than just one. The image is also more efficient than the 3D fitness landscape, in that the perception of a 3D image requires the recognition of the third spatial dimension.

Providing a high-dimensional to low-dimensional mapping could be produced, such that the location of the chromosomes could be used to identify the chromosomes' values. The user could see not only the population's diversity, convergence, and the clustering of chromosomes, but also the relationship between the local structure of the chromosomes and their fitness ratings. Hence, a single visualization could be used to answer all three of the user's anticipated queries.

5.3.3 A Set of Applicable Visualization Techniques

Visualizations of high dimensional search spaces rely on the effective "translation" or "mapping" of each point in the high dimensional search space (i.e. each chromosome) to a unique point in a

low dimensional plot. This section identifies two applicable mapping techniques for achieving this goal: “Sammon Mapping” and “Extensive Repartitions,” and explains why the extensive repartitions technique is considered superior for visualizing GA search spaces.

Sammon Mapping

Sammon mapping is a technique for mapping high dimensional data to fewer dimensions whilst preserving the Euclidean distances between the data points [Sammon, 1969]. High dimensional GA problem spaces can be represented in r dimensions using this technique¹. Sammon mapping is a non-linear technique that starts with a random set of 2D points and iteratively reduces the error between the Euclidean distances separating the mapped points in the 2D representation and the Euclidean distances separating the data points in the high-dimensional space. The error associated with a mapping at iteration m is given by:

$$Error_{\langle m \rangle} = \frac{1}{\sum_{i < j} [d_{ij}^*]} \sum_{i < j} [d_{ij}^* - d_{ij\langle m \rangle}]^2 / d_{ij}^*,$$

where, d_{ij}^* denotes the Euclidean distance between the i -th and j -th chromosomes in the high-dimensional search space and $d_{ij\langle m \rangle}$ is the Euclidean distance between the corresponding coordinate pair in the 2D map after the m -th iteration. [Dybowski et al., 1996] used a steepest-descent procedure for minimizing the $Error_{\langle m \rangle}$, given by:

$$v_{ik\langle m+1 \rangle} = v_{ik\langle m \rangle} - \frac{\zeta}{\left| \frac{\delta^2 Error_{\langle m \rangle}}{\delta v_{ik\langle m \rangle}^2} \right|} \cdot \frac{\delta Error_{\langle m \rangle}}{\delta v_{ik\langle m \rangle}},$$

where, ζ is an empirically derived constant (typically about 0.4) associated with the step length. Figure 5.6 shows an illustrative example of a Sammon map for a 5 bit binary problem space. From a complete set of mapped 2D coordinates (i.e. a pre-defined look-up table) any set of chromosomes can be identified and highlighted as a set of points on a 2D scatterplot (see Figure 5.7).

The use of Sammon mapping for showing a GA’s search behaviour should be constrained to binary problem representations. It is proposed that the use of Euclidean distance measures is inappropriate

¹This technique was first applied to EC as part of this project in collaboration with Richard Dybowski from the Microbiology Department, St Thomas’ Hospital, London and Peter Weller at City University, London. A detailed description of this technique was presented at the 1996 Annual Conference on Evolutionary Programming, see [Dybowski et al., 1996].

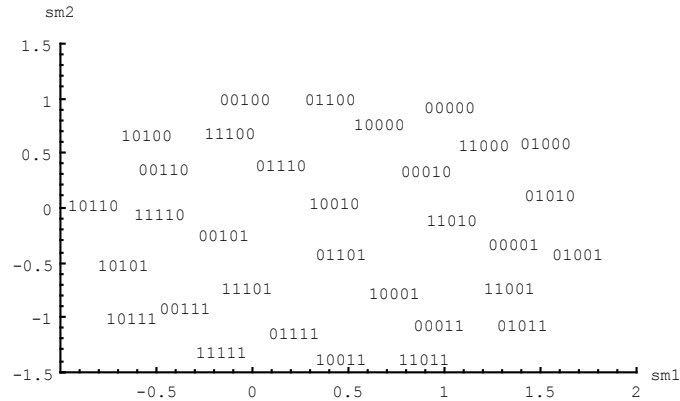


Figure 5.6: A Sammon map produced from a dataset of all the combinations of a 5 bit binary string. This mapping attempts to preserve the Euclidean distance between the points in the high (i.e. 5) dimensional search space and the two dimensional scatterplot representation. This is most effective for items around the edge of the 2D scatterplot.

for representation schemes with more than two alleles: The magnitude of the allele differences determine the location of the chromosomes on the 2D map - yet the magnitude of the allele differences may not be indicative of the differences between the alleles in the genotypic search space. For example, a 4 bit decimal chromosome of 9000 would be placed much further away from 0000 than 1111 would be, yet in terms of its genotypic traits 9000 is more similar to 0000 than 1111 is to 0000.

Extensive Repartitions

Extensive repartitions repeatedly partition an axis in order to illustrate classifications. One of the first reported uses of extensive repartitions was by Charles De Fourcroy in 1782, a Director of Fortifications in Paris, France to illustrate the relative population sizes of French cities (see [Bertin, 1983, page 203]). This technique has more recently been applied as a multi-variate data visualization technique, see [Mihalisin et al., 1991].

Search space matrices (“SSM”) apply extensive repartitions in order to construct low dimensional representations of a GA’s high dimensional search space [Collins, 1996], [Collins, 1997] and [Collins, 1998]. Rather than scaling or mapping the population data to produce scatterplots, a matrix can be used to illustrate the entire search space. This is achieved by constructing the matrix elements an allele at a time, partitioning the axes of the matrix horizontally and vertically, and filling

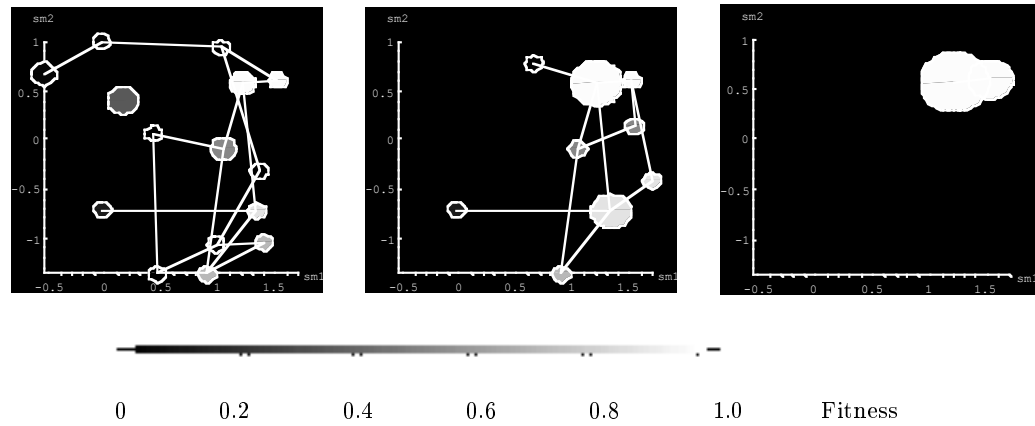


Figure 5.7: A search space visualization of the progress of a GA one a problem with a single optimum solution. The chromosomes from generation 0, 2 and 20 are shown in the above three images, each circle illustrates a chromosome, the position of each circle illustrates the chromosome's location in a 2D Sammon map, the size of each circle illustrates the frequency of that chromosome in the population and the value image variable illustrates the chromosome's fitness rating. A line linking all the Hamming 1 neighbours of the fittest chromosome is shown in each image.

each subsection with alternate alleles. Figure 5.8 shows an illustrative example of the construction of a 2D search space matrix for a four bit binary search space.

Rather than generating a matrix of the entire search space, the following direct linear equation can be used to translate a specific chromosome to its coordinate in r dimensions (r is typically 2 or 3):

$$\begin{aligned} \text{chromosome coordinate} &= \sum_{i=d}^{i \geq p} X_i \times W_i \\ \text{in dimension } d & \end{aligned}$$

where i is incremented by r , X is a chromosome with p variables, X_i is the position of each allele in the set of possible alleles indexed from the right at $i = 1$. Each locus i has B_i different alleles, and W_i is the contributing weight of each locus, where $W_{1...r} = 1$ and $W_{(r+1)...p} = W_{(i-r)} \times B_{(i-r)}$. For example, the binary string $X_i = (0110)$ maps to the 2D coordinate position $(2, 1)$. this is calculated as follows: a four bit binary string has a base list $B_i = (2222)$, and a list of weights $W_i = (2211)$, hence $x = (0 \times 1) + (1 \times 2) = 2$ and $y = (1 \times 1) + (0 \times 2) = 1$.

Using the extensive repartitions technique not only can p bit binary strings be shown in r dimensions, but also any other form of categorical data. Figure 5.9 shows an example search space matrix

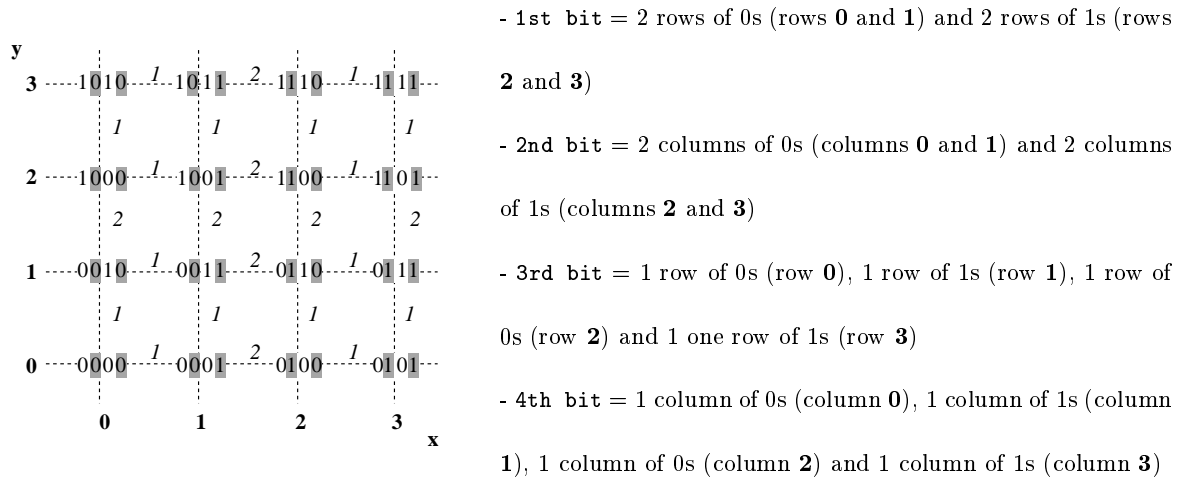


Figure 5.8: The construction of a complete search space matrix of a 4 bit binary dataset. The chromosomes at each position in the search space are shown on the right hand side, background shading is used to highlight the loci of the same alleles and italics are used to indicate the Hamming distance between neighbouring chromosomes.

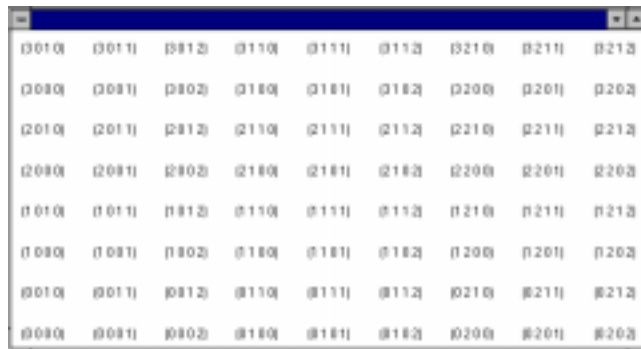


Figure 5.9: A search space matrix illustrating the complete search space for a 4 bit coding alphabet of (0 1 2 3), (0 1 2), (0 1) and (0 1 2).

for a GA with a four bit coding alphabet of (0 1 2 3), (0 1 2), (0 1) and (0 1 2).

Sammon Mapping vs Extensive Repartitions

As noted at the beginning of this subsection the visualization of high dimensional search spaces relies on the *effective* mapping of each chromosome to a unique point in a low (2 or 3) dimensional plot. The effectiveness of the above two methods is discussed here in terms of the effort required to produce the mapping and the accuracy of the end result. The effort required to produce the mapping is judged in terms of the computational complexity of the mapping technique, accuracy is analysed in terms of the error associated with the relative interpoint distances in the high and low dimensional spaces.

Table 5.3: A comparison of the computational complexity (expressed in big- O notation [Aho and Ullman, 1992]) of principal component analysis, Sammon mapping, Kohonen feature mapping and Bishop's latent-variable mapping.

This table was taken from [Dybowski et al., 1996, page 382].

TYPE OF MAPPING	COMPUTATIONAL COMPLEXITY	
	In general	For all possible binary chromosomes of length p , i.e. $n = 2^p$
PCA	$O(\max(np^2, p^3))$	$O(2^p p^2)$
Sammon	$O(n(n+1)(p+m+1)+m)$ for constant r	$O(2^{2p}(p+m))$ for constant r
Kohonen	$O(n(n_{array}(p+1)+1)(m+1)+m)$, where $n_{array} \geq n$	$O(2^{2p}p(m+1))$ if $n_{array} = n$
Bishop	$O(Kp(n+M))$ for constant m	$O(K2^p p)$ for constant m, M

where,

n : number of data points

n_{array} : number of array points

p : dimension of data space

r : dimension of mapping codomain (equal to 2 or 3)

m : number of iterations

K : number of kernel functions in mixture model

M : number of basis functions in generalized linear neural network

Computational Complexity

[Dybowski et al., 1996] explored the computational complexities of Principal Component Analysis ("PCA"), Kohonen feature mapping [Kohonen, 1989] and Bishop's latent-variable mapping [Bishop, 1995], when used to map complete GA search spaces. The computational complexity of mapping a complete search space for p bit binary chromosomes (i.e. $n = 2^p$) rises exponentially with respect to the number of points n for all four mapping techniques (see Table 5.3, central column). In the case of Sammon mapping the exponential rise in computational complexity is quadratic with respect to the number of points being mapped (see Table 5.3, right column).

The mapping function used by the extensive repartitions technique to translate a chromosome to a low-dimensional coordinate on the other hand rises linearly with respect to the length of the chromosome being mapped. The mapping does not need to be made for the entire search space as

Table 5.4: Sammon Mapping vs Extensive Repartitions: A comparison of the total error value for a complete search space mapping and the mean error value for each mapped datapoint is given here. The values listed for the Sammon mapping method are the average values taken from ten 100 iteration runs of the Sammon mapping algorithm.

MAPPING	Sammon Map	Search Space Matrix
TOTAL $Error_{\langle m \rangle}(q)$	23.0307	19.0301
MEAN $Error_{\langle m \rangle}(q)$	0.395854	0.297345

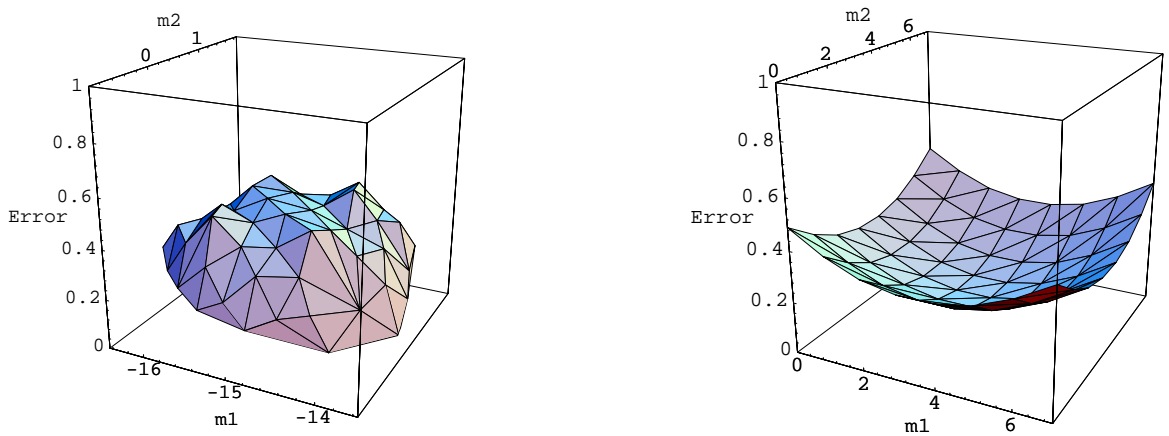


Figure 5.10: Two error surface plots for a Sammon mapping after 100 iterations from a random initial configuration (left) and an extensive repartitions mapping (right) of a six bit binary search space.

the mapping function will produce a unique coordinate for each chromosome it is applied to.

Accuracy

The accuracy of the two mapping techniques for binary search spaces can be investigated by comparing the total Euclidean based error for mappings produced by each technique and examining the variation of the error rating across the low dimensional mapping. A 3D surface plot of the error surface i.e. a 3D plot in which the error rating is illustrated on the y axis and the two mapped dimensions are shown on the x and z axes, supports this comparison. Table 5.4 presents the total and average Euclidean error values for a Sammon mapping and extensive repartitions mapping of a six bit binary search space. Figure 5.10 shows the error surface for both mapping techniques.

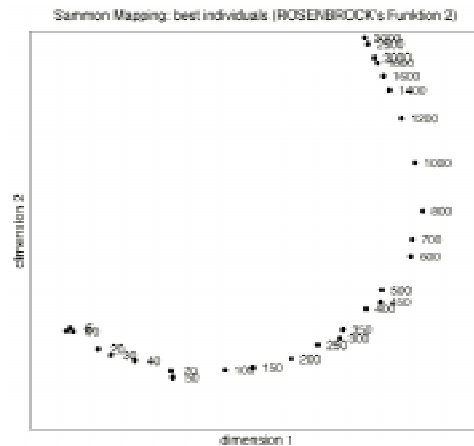


Figure 5.11: A point plot showing the best individual in a GA's population at a number of generations during the GA's run. Rather than mapping the entire search space, the Sammon map used here was produced by mapping only the chromosomes considered by the GA during its run. This figure was taken from [Pohlheim, 1998].

As can be seen in Table 5.4 the total error for a mapping produced by the extensive repartitions technique is less than that produced by Sammon mapping. Although only a six bit binary mapping is given here to illustrate the point, this finding is characteristic of the base number of the data set (i.e. binary data) and is independent of the size of the search space being mapped. Furthermore, as shown in Figure 5.10 the distribution of the points produced by Sammon mapping is quite distinct from that produced by the extensive repartitions technique. Extensive repartitions form a bowl shaped error surface with the maximum error values occurring around the outside edge, rather than the central region as found for the Sammon mapping.

Conclusion - Extensive Repartitions

Of the two methods described above the extensive repartitions technique is the better method for producing search space visualizations. The extensive repartitions technique is more accurate and easier to apply than the Sammon mapping technique. It is for these reasons that the extensive repartitions technique is used in the remainder of this thesis to produce the example search space visualizations. The same visualizations could be produced by Sammon mapping through the use of a look-up table containing the GA's chromosomes and corresponding coordinate data.

Aside - Search Sample Visualization

One response to the problem of computational complexity is to limit the mapping to a subset of chromosomes in the search space. For example, only mapping those chromosomes considered by the GA. Based on the findings of [Dybowski et al., 1996] Hartmut Pohlheim took up this approach and explored the use of Sammon mapping to produce 2D scatterplots of the search sample [Pohlheim, 1998] (see Figure 5.11).

Mapping only the chromosomes considered by the GA greatly reduces the number of points to be mapped and therefore, the time taken to produce a Sammon map. However, there is a cost; the Sammon map can only show the chromosomes included in its construction. Hence, the separation between chromosomes in the Sammon map indicates the relative Euclidean distances between the chromosomes in the search sample rather than the absolute differences between the chromosomes, and the Sammon map shows the population's diversity in the search sample rather than the population's diversity in the problem space. As a result, the user is unable to see the population's diversity or convergence in the search space, and no direct relationship can be derived between the regions of the search sample visualization and the local structure of the chromosomes they contain.

Alternate Points of View from Alternate Partitions

Alternate view points can be produced with the extensive repartitions technique by changing the weighting function W_i . For example, a Gray coded weighting function can be used instead of a normally summed weighting function. This uses a Gray code for to calculate the values of the weights W_i in the coordinate translation function. In this case the weight value for each locus depends on the value of the previous loci, for example Gray 00 = decimal 0, Gray 01 = decimal 1, Gray 11 = decimal 2, Gray 10 = decimal 4. In the case of a Gray coded weighting function the chromosomes are distributed such that the Hamming distances (i.e. the numbers of different bits) between all of the neighbouring chromosomes is 1 (see Figure 5.12).

The example mappings given in the remainder of this thesis use extensive repartitions based on normal base summed weights rather than Gray coded weights. Although both techniques are equally applicable, the original technique is used as the resulting distribution of the points is easier to remember - a string of all 0s appears at the bottom left hand corner, a string of all 1s appears at

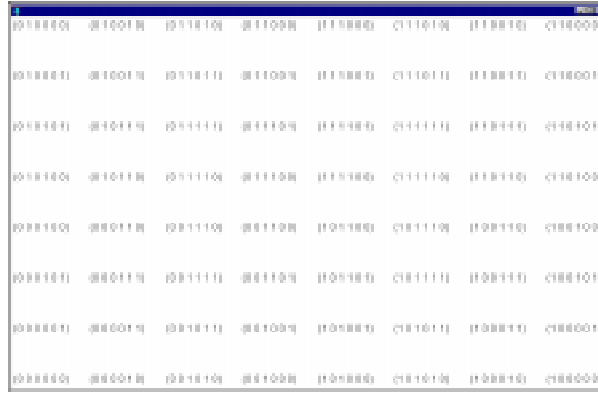


Figure 5.12: A 6 bit binary search space matrix produced using a Gray coded weighting function.

the top right hand corner and strings of alternate 0s and 1s appear at the top left and top right hand corners (this distribution is independent of the length of the string).

Given that the search space matrix is produced by repartitioning the display space, the values at the start of the chromosomes will be shown in fewer subdivisions than those at the end of the chromosomes. Therefore, another way of altering the user's viewpoint is to re-order the application of the weighting function for each locus so that different projections can be made in order to emphasize different patterns in the population. For example, if the chromosomes represent a single binary number then the search space should be partitioned in such a way that the alleles at the start of the chromosome are represented in the first few partitions, as shown in the above examples. Similarly, if the chromosomes show more than one gene then the partitioning should be ordered from the most to least significant bit of each gene.

Finally, the fact that a search space matrix is made up of alternate symbols across its rows and columns means that it can be used to emphasize the schema construction of the chromosomes it contains, for example all of the chromosomes shown in the bottom half of a binary search space matrix contain a 0 in their first locus, and all of the chromosomes shown on the left hand side of a binary search space matrix contain a 0 in their second locus. As a result, a dialog in which the user can identify a schema of interest may be used to highlight the rows and columns on which these schema occur (see Figure 5.13). By enabling the user to highlight regions of the search space containing a schemata of interest, the user can seek out schema that emerge during evolution and directly examine the local structure of the chromosomes and the impact the local structure has on

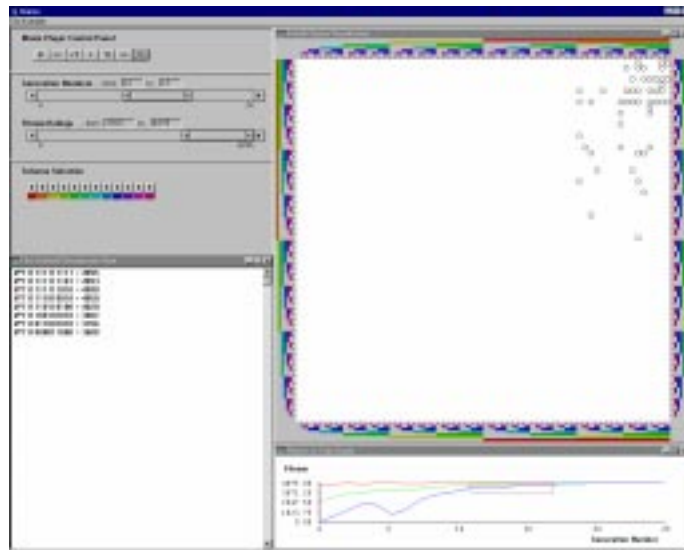


Figure 5.13: An example GONZO screen view illustrating how a search space visualization (top right) can be used to examine the structure of a specific chromosome 111111111111 (middle left). Each colour ribbon in the search space visualization (top right) identifies the regions of the search space that contain chromosomes with a 1 at the corresponding colour ribbon's locus, as shown in the Schema Selection dialog (middle left). The individual components of this screen view are explained in detail in Chapter 7.

the chromosomes' fitness ratings.

5.3.4 Discussion

Exploring suitable mapping techniques for scaling the high dimensional search space of the GA to the 2D display space of the screen was the most difficult task in this project. Scaling techniques such as PCA and biplots rely on the ability of the scaled components and factors to show the majority of the diversity in the data, but for a complete search space of every possible combination of points this is no longer possible as the data are distributed evenly over every variable (i.e. every chromosome locus). As a result these scaling techniques are not applicable for showing an entire search space. However, PCA has been applied to map the points in the search space sampled during a GA's run [Harvey and Thompson, 1996]. This shows the clusters within the chromosomes considered during the GA's run (see Figure 5.14). Although this approach is faster than Sammon mapping, like the approach proposed by Hartmut Pohlheim (see Section 5.3.3) it can only show the chromosomes considered during its creation, the results of different runs cannot be compared unless a complete

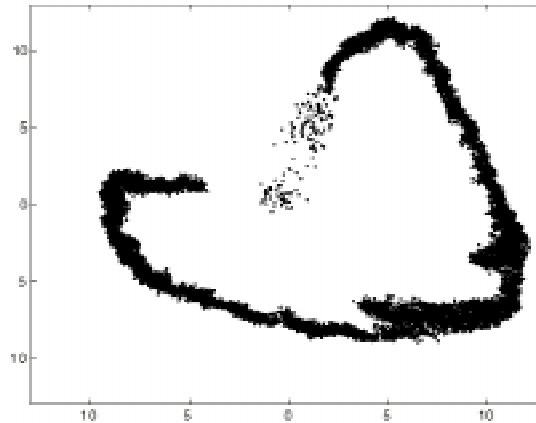


Figure 5.14: A point plot showing all fifty individuals in a GA's population for every tenth generation of the GA's 5220 generation run. This view uses the first two principal components of a principal component analysis of all of the points to be shown, to plot each chromosome at a single 2D point. The initial population in this case appears as a scattering of points at the centre of the view (at 0,0) and slowly converges, whilst moving in a clockwise direction, to the cluster shown at the end of the trail to the left of the origin (at -5,0).

mapping of all the chromosomes is made, and the relative position of each chromosome shows its diversity in the search sample rather than the search space.

For iterative scaling techniques like Sammon mapping, that rely on maintaining the relative Euclidean distances between the points in the high and the low dimensional spaces, the computational complexity of the technique can be dissuasive. However, once produced the mapping can be recorded and fast indexing techniques can be applied to access either the 2D coordinate values for any given chromosome, or the chromosome for any given 2D coordinate. So although producing an accurate representation of the search space by iterative scaling techniques can be time consuming once created the mapping can be re-used whenever required.

With the extensive repartitions technique the mapping itself can be optimized to the resolution of the current display. If the first twelve bits are the only bits that determine the unique pixel location of any given chromosome, then it is unnecessary to map the entire string, rather the mapping technique should be applied only to those loci that contribute. For example, given a 20 bit binary chromosome in which only the first 12 bits determine unique pixel positions, only the first 12 bits (loci 0 to 11) would be used to produce the mapping. However, if the user zoomed into the second half of the display area then the alleles at loci 12 to 23 would be used. If they zoomed further into one quarter of

the search space then the alleles at loci 2 to 13 would be used, and so on. In this way the complexity of the mapping is bounded by the resolution of the screen display area.

If the user is interested in the local structure of specific regions of the search space then the efficacy of the extensive repartitions technique far exceeds any of the known alternatives. Sammon mapping will organize the chromosomes by Euclidean distance, whereas extensive repartitions will organize the display based on the division of the space into sections (where the number of sections is determined by the number of alleles at each locus). Using additional navigational support enables the user to highlight sections of the partitioned search space representation in order to reinforce the user's perception of the chromosomes' local structure.

Furthermore, given that the ordering of the alleles in the chromosome determines the projection of the points in the 2D search space view, the user can also investigate the formation of building blocks (i.e. fit schemata) by changing the locus partitioning order to group the alleles from different loci. The second order schemata view suggested in [Spears, 1994] is an effective way of illustrating the frequency of second order schemata (see Section 4.2.1, Figure 4.23 on page 109), but with the search space matrix the user can re-order the locus projection such that any schemata can be identified as a region of the search space view. The individual chromosomes that contain that schemata, along with their fitness ratings, will then appear within the identified region of the search space view.

5.3.5 Visualization Summary

This section presented an approach to visualization design based on the principles of the graphic sign system and illustrated how this approach could be applied to visualize a GA's search space. Three 'common' questions were extracted from the responses given in the GA user study:

- How diverse/converged are the chromosomes in the population?
- Are there clusters of chromosomes forming during the GA's run?
- How does the local structure of the chromosomes affect the chromosomes' fitness ratings?

Of the visualizations available at the start of this project only the fitness landscape visualizations were effective for answering these questions. However, the fitness landscape visualizations were only applied to show one or two problem dimensions. Two applicable mapping techniques, Sammon

Mapping and Extensive Repartitions, were examined for producing two dimensional visualizations of the search space of GAs. Table 5.5 summarizes the features of Sammon mapping and search space matrix visualizations.

Sammon mapping is an example of an iterative error-reduction approach for mapping a number of data points in a high dimensional data set to (the same number of) points in a fewer number of dimensions, whilst preserving the relative Euclidean distances between each data case. Because this approach relies on reducing the error associated with the Euclidean distances between all of the points in the high and low dimensional spaces, the computational complexity of this approach rises quadratically with respect to the number of cases.

Search Space Matrices are a GA visualization technique that use *extensive repartitions* to directly translate a chromosome in the high dimensional search space to an r dimensional coordinate position (where r is typically 2 or 3). The computational complexity associated with search space matrices rises linearly with respect to the length of the chromosomes. Search space matrices are more accurate and easier to produce than Sammon mappings, and are used in the remainder of this thesis to illustrate the use of search space visualization.

Table 5.5: A summary of the advantages and disadvantages of using Sammon Mapping and Search Space Matrices to visualize the search behaviour of a GA.

	SAMMON MAPPING	SEARCH SPACE MATRICES
<i>Producing the map</i>	Maps a complete set of points.	Maps each point independently.
	The computational complexity of the mapping rises quadratically with respect to the number of points.	The computational complexity rises linearly with respect to the length of the chromosome.
	The entire search space must be mapped before any visualizations are produced.	Chromosomes can be mapped “on the fly” and the form of projection altered at will.
<i>Using the mapping</i>	Storing the map as a look up table enables the quick production of views, and can be used to access coordinates from chromosomes or access chromosomes from coordinates.	The search space matrix can be stored and used in a similar manner to the Sammon map, or the translation function can be applied on the fly to produce the visualization.
		The structure of the search space matrix can be used to identify the local structure of the chromosomes in the 2D search space.

Chapter 6

HENSON: A GA Visualization

Framework

As identified in the last chapter (Section 5.1), the provision of an extendable framework for developing GA visualizations is essential, in order to support both the intuitive development of existing visualization designs (as described in Section 4.1), and the user’s future development of as yet unconsidered visualizations. This chapter presents “HENSON,” an extendable GA visualization framework. HENSON is based on the VIZ SV development framework described in Section 4.2.2 but is extended here to support the development of GA specific visualizations. The following three extensions are made to the VIZ framework:

1. A set of standard GA *players* and *events* are identified.
2. A series of GA specific *views* are introduced into the framework’s view hierarchy.
3. An extension is made to the VIZ architecture to enable the user to edit components of the underlying GA, such as the algorithm’s genetic operators and parameters.

Section 6.1 describes a generic execution model for GAs and (based on the generic execution model) proposes a set of standard GA *players* and *events* suitable for describing the GA’s operation at a number of different levels of abstraction. Section 6.2 illustrates the standard VIZ view hierarchy and identifies the additional views required in order to produce the existing GA visualizations reviewed

in Section 4.1. Thirdly, Section 6.3 describes the necessary changes that need to be made to the architecture of the VIZ framework in order to enable the user to edit the underlying algorithm. Finally, Section 6.4 presents an illustrative example of how HENSON can be applied to produce a visualization of the population's chromosomes and fitness ratings from a GA run.

6.1 Generic GA Players and Events

A range of reusable GA implementations are available that support the development of GAs (see [Ribeiro et al., 1994]). Many of these systems constrain or bias the user towards a particular GA approach or problem domain, but generic systems such as GALIB [Wall, 1996] or GECO [Williams, 1993] enable an unbiased approach. The generic data structures and execution model used in GECO is adopted here in order to identify a set of generic GA players and events suitable for producing a wide range of GA visualizations. The generic data structures and execution model used in GECO is comparable to those used in a range of other generic GA frameworks (including GALIB) and is consistent with those identified by the respondents of the GA user study.

“GECO” stands for the Genetic Evolution through Combination of Objects and is a CLOS-based¹ framework for prototyping GAs [Williams, 1993]. Although designed for prototyping GAs, Williams considers it to be sufficiently flexible to be used for prototyping learning classifier systems, genetic programming algorithms [Koza, 1992], ESs and EP [Williams, 1993, pages 4 and 5]. GECO is an object oriented library which implements an environment primarily in the form of classes and methods. The principal classes form a hierarchy of abstractions (as opposed to a class inheritance hierarchy) that describes the concepts of genetic evolution, specifically the “ecosystem,” “population,” “organism” and “chromosome.”

6.1.1 Generic GA Players

The GECO abstraction hierarchy identifies the primary data structures in all GAs. The *ecosystem* is the highest level of abstraction in a GECO implementation, and it refers to both the population undergoing evolution and the genetic plan which control the evolution. GAs evolve *populations* of

¹CLOS stands for the Common Lisp Object System which is an object oriented version of Lisp.

Table 6.1: The HENSON GA specific players associated with a GA's principal class hierarchy.

PLAYER	SLOTS	DESCRIPTION
plan	ecosystem generation-limit evaluation-limit	The genetic plan used by the GA, including slots for the ecosystem to which it belongs, and the maximum number of generations and evaluations to be carried out before termination.
ecosystem	population generation-number evaluation-number plan	The highest level player providing a handle on the entire GA and its run, ecosystem identifies slots for the population, current generation number, the number of evaluations done so far, and the overall genetic plan being applied.
population	ecosystem organisms size statistics	An instance of the population class that maintains a link back to the ecosystem that contains it, as well as slots for a vector of all the organisms in the population, the size of the population, and an instance of the population statistics class.
organism	population genotype phenotype score normalized-score	Each organism instance contains a link back to the population, the genotype contains a list of chromosomes, the phenotype provides an explicit phenotypic representation of the organism, the score contains the organism's raw numeric fitness rating, and the normalized-score contains a normalized version of score with respect to the rest of the population.

organisms, where the current population at any time is the set of organisms that interact with one another to produce new organisms. An *organism* combines all the information relating to a single structure in the search space of a GA. An organism is a member of a population and generally has a coded genetic description i.e. its “genotype” which interacts with the environment as the organism’s “phenotype.” The *chromosome* is a structured component of an organism’s genotype and is generally the unit operated upon by the GA’s genetic operators. Each chromosome is generally composed of a vector of loci (sites) each of which may take on one set of values (i.e. the values for that locus). These four primary data structures along with two specialized chromosome sub classes, the **genetic-plan** and **population-statistics** classes make up the set of generic GA *players* in HENSON. Tables 6.1, 6.2 and 6.3 describes the standard players used in HENSON and identifies their slot values.

Table 6.2: The HENSON GA specific players associated with the details of the individual organisms.

PLAYER	SLOTS	DESCRIPTION
genotype	chromosome	A list of one or more chromosomes used to represent the GA's possible solutions to the given problem.
phenotype		An explicit phenotypic representation of the organism.
score		The fitness rating of the organism.
normalized-score		The normalized fitness rating of the organisms with respect to the maximum and minimum fitness ratings of organisms in the population.
chromosome	organism loci	The chromosome hold a link back to organism and a loci-vector encoding the genetic information of the chromosome.
loci		The loci is used to encode the genetic information of the the chromosome.
binary-chromosome		A specialist sub class of chromosome used to represent binary chromosomes.
sequence-chromosome		A specialist sub class of chromosome used to represent sequence based chromosomes.

Note, potentially any structure in any program could be a player. A player is an object of interest used to produce a visualization; in HENSON the generic players identified above are the primary data structures that a GA manipulates during execution. These are the anticipated objects of interest for producing GA visualizations. Although the user may introduce specialist sub classes, the generic players identified here will still be applicable, and like the view hierarchy, the user is able to extend their set of GA specific players at anytime.

6.1.2 Generic GA Events

This subsection describes the visualization events for GAs, which follow directly from the generic execution model. The execution model used in GECO is as follows:

Initialization: Make an instance of the ecosystem class or subclass which will be used for the GA.

Table 6.3: The HENSON GA specific players associated with the statistical data of the GA's population.

PLAYER	SLOTS	DESCRIPTION
population- statistics	population sum-score avg-score max-score min-score max-organism min-organism sum-normalized-score avg-normalized-score	The extendable set of population statistics recorded per generation, with slots for a link back to the population, as well as the sum, average, maximum and minimum of all the scores in the population, plus the organism in the population that had the maximum score, the organism that had the minimum score, the sum of all the normalized scores, and the average of all the normalized scores.
sum-score		The sum of the chromosomes' fitness ratings in a population.
avg-score		The average chromosome fitness rating in a population.
max-score		The maximum chromosome fitness rating in a population.
min-score		The minimum chromosome fitness rating in a population.
max-organism	population genotype phenotype score normalized-score	The organism with the maximum fitness rating in a population.
min-organism	population genotype phenotype score normalized-score	The organism with the minimum fitness rating in a population.
sum- normalized- score		The sum of the chromosomes' normalized fitness ratings in a population.
avg- normalized- score		The average of the chromosomes' normalized fitness ratings in a population.

Table 6.4: The HENSON GA specific initialization events.

EVENT	PLAYERS	DESCRIPTION
make-genetic-plan	ecosystem	Creates the genetic-plan used in GA's evolution.
make-population	ecosystem	Creates the GA's initial population including the individual organisms.
make-organism	population	Used by the make-population event to create each organism in the initial population.
make-loci-vector	chromosome	Used by the make-organism event to create each chromosome's loci-vector.

Evolution: Invoking `evolve` on the ecosystem causes `GECO` to evolve the population. This consists of repeating the following steps:

- Evaluate each of the organisms in the current population, recording a `score` for each one.
- Calculate the population statistics, normalized `scores` for each organism, and normalized population statistics.
- Determine if the GA's termination condition has been met. If it has then terminate. Otherwise:
 - **Regenerate** the population. This typically includes selecting members of the previous population and applying reproduction operators such as crossover or mutation to create the members of the new population.
 - Recursively evolve the result.

This model presents a very generic description of evolution and identifies the basic flow of control in a generic GA. The main algorithm components are the initialization and evolution stages in the algorithm. Within the evolution stage the “evaluate organisms,” “calculate statistics,” “test termination condition” and “regenerate the population” steps characterize the GA's evolution. This generic model is sufficiently general to describe the execution of practically any GA. In order to define a specific GA design, instances of these classes and specialist sub classes are used.

Table 6.5: The HENSON GA specific evolution events.

EVENT	PLAYERS	DESCRIPTION
<code>evaluate</code>	<code>ecosystem</code>	Evaluates each organism in the ecosystem's population according to the ecosystem's genetic plan and records the result in the organism's score slot.
<code>evolution-termination-p</code>	<code>ecosystem</code>	Checks to see if the termination conditions are satisfied e.g. if the maximum number of generations or evaluations has been exceeded, or if the population has completely converged.
<code>regenerate</code>	<code>ecosystem</code>	Calls the ecosystem's regenerate method to evolve the current population according to the ecosystem's genetic plan and record the old population's statistics in a list of statistics for the genetic plan.

The generic execution model identifies two parts of the GA's execution, the "initialization" and "evolution" of the algorithm. These are the two most abstract algorithm events, the individual events involved in the high-level *initialization* event are described in Table 6.4. The `make-genetic-plan` and `make-population` events create the instances of the `genetic-plan` and `population` classes, respectively. The `genetic-plan` includes the methods associated with the organisms' evaluation, algorithm termination, organism selection, and reproduction. The `make-population` event creates an instance of the `population` class which in turn repeatedly calls the `make-organism` event to create the organisms in the initial population, which applies the `make-loci-vector` event to create the loci-vector for each locus slot of the organisms' chromosomes.

The *evolution* events are presented in Table 6.5. A GA's evolution breaks down into three stages: the evaluation of the organisms in the population, a test to verify if the algorithm should stop or continue, and the regeneration of a new population from the old population based on the genetic-plan being applied. The `evaluate`, `evolution-termination-p` and `regenerate` events reflect this three stage process. The `regenerate` event includes the selection and reproduction of the population's organisms. Tables 6.6 and 6.7 identify the selection, crossover and mutation methods included in

Table 6.6: The HENSON GA specific regenerate *selection* events used in the GA's evolution.

EXAMPLE EVENTS	DESCRIPTION
<code>pick-random-organism</code>	Returns a random organism from the population.
<code>roulette-pick-random-organism</code>	Selects random organisms from the population, weighted by score, using the roulette wheel approach (see [Goldberg, 1989], or Section 2.2.3).
<code>stochastic-remainder-preselect</code>	Selects random organisms from the population weighted by score, using Brindle's stochastic remainder selection without replacement (see [Brindle, 1981]).
<code>ranking-preselect</code>	Selects a random organism from the population, weighted by the rank of each organism's score within the population.
<code>tournament-select-organism</code>	Picks a number of organisms from the population at random and returns the best one.

GECO; identifying each of the sub events is necessary when producing operator specific visualizations, for example visualizing roulette wheel selection.

6.2 The HENSON GA View Hierarchy

VIZ views are used to produce graphical structures such as 2D line graphs, point plots and trees. These are arranged in an object-oriented inheritance hierarchy (see Figure 6.1). This enables the user of the framework to produce new views by inheriting the methods of a more general view and adding the necessary specializations. Assuming that any visualization system can provide a complete set of views for all of its users needs is unrealistic. Their needs will change as their understanding of the subject changes, the area of application changes, and the visualization field itself changes. The

Table 6.7: The HENSON GA specific regenerate *reproduction* events used in the GA's evolution.

STEP	EXAMPLE EVENTS	DESCRIPTION
<i>Crossover</i>	cross-organisms	Performs a simple crossover between two parent organisms, producing two child organisms.
	uniform-cross-organisms	Performs uniform-crossover on two parent organisms producing two child organisms (see [Syswerda, 1989] or [Davis, 1991]).
	2x-cross-organisms	Performs two point crossover on two parent organisms to produce two child organisms.
	r3-cross-organisms	Performs the random respectful recombination crossover operator between two parent organisms, resulting in two child organisms (see [Radcliffe, 1992]).
	pmx-cross-organisms	Applies partially mapped crossover (PMX) between two parent organisms resulting in two child organisms (see [Goldberg, 1989]).
<i>Mutation</i>	mutate-organism	Mutates an organism randomly.

provision of an extendable hierarchy of views is therefore considered an essential part of a pragmatic approach to visualization.

VIZ was developed to facilitate the visualization of the processes involved in knowledge engineering (see [Domingue et al., 1993]). The majority of views used by knowledge engineers at that time were 2D plots and graphs, such as histograms, line graphs, point plots, pie charts, dials, tables, pretty-printed code listings and trees. Hence, these were the default set of views provided for within VIZ.

The intended area of application for HENSON is GAs, so the default set of views should reflect the anticipated needs of the GA community. Fortunately the existing body of research on GA visualization has produced a set of visualizations that can be used here as a starting point (see Section 4.2.1). In order to provide support for these visualizations the base set of VIZ views must be extended the

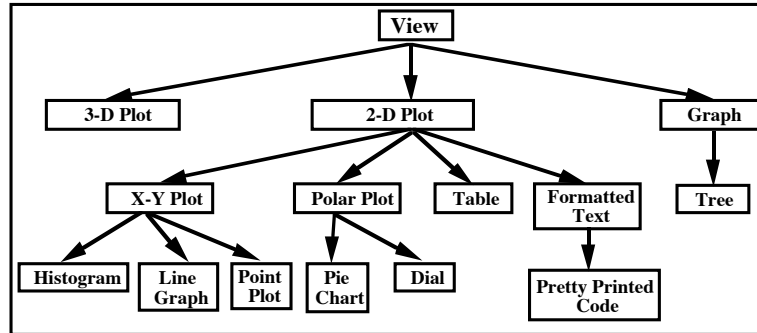


Figure 6.1: The inheritance hierarchy of views available in VIZ. This figure was taken from [Domingue et al., 1993, page 12].

resulting view hierarchy for HENSON is shown in Figure 6.2.

A set of 3D plots is introduced to support the development of 3D fitness graphs as used in [Harvey and Thompson, 1996], 3D search space plots as used in [Nassersharif et al., 1994] and [Spears, 1994], and 3D fitness surfaces as shown in [De Jong, 1975] and [Spears, 1994]. A specialist 2D graph view is introduced to support the visualization of graph based phenotype views, specifically for the traveling salesperson problem, as used in [Stasko, 1989] and [Dabs and Schoof, 1995]. A range of 2D histogram, 2D line graph, 2D point plot and polar plot views are created to support a range of fitness oriented visualizations (see [Kapsalis et al., 1993], [Collins, 1993] and [Dabs and Schoof, 1995]) and chromosome summary visualizations (see [Collins, 1993] and [Spears, 1994]). Finally, an icon view is introduced for producing chromosome icons as proposed in [Collins, 1993], [Spears, 1994] and [Wu et al., 1998].

6.3 The HENSON Architecture

The main difference between the architecture of VIZ and HENSON is the additional link made between the navigator module and the annotated source (see Figure 6.3 on page 171). This enables the user to edit the underlying source of the visualization - for example, to make changes to the GA's parameters or algorithm components.

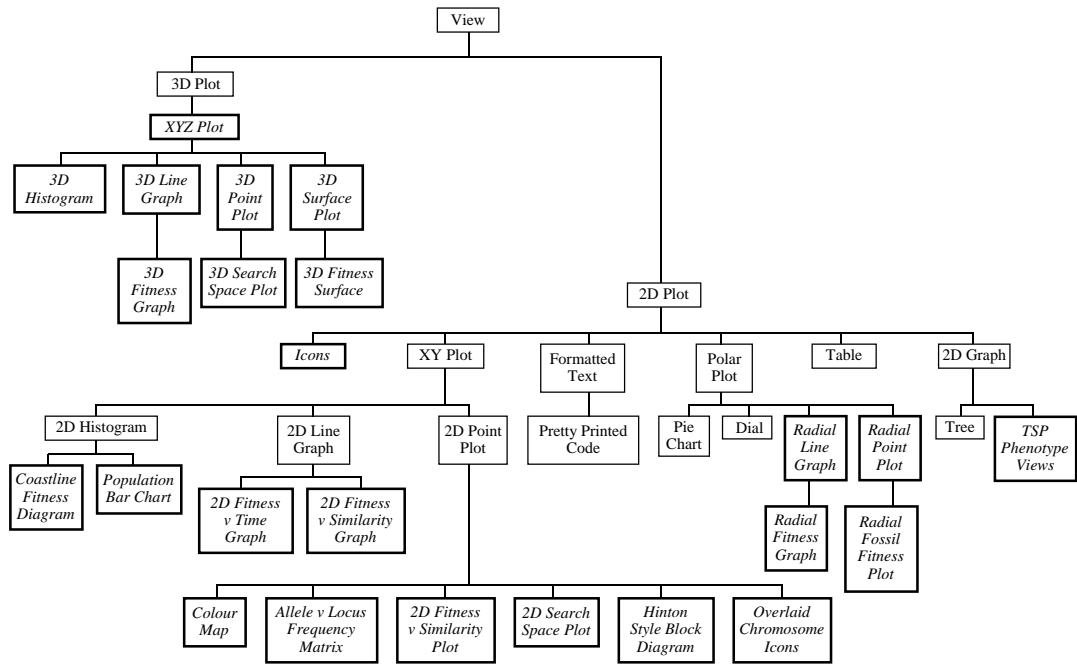


Figure 6.2: The HENSON view hierarchy. The extensions to the VIZ view hierarchy are shown here in bold boxes with italicised labels.

6.4 Example GA Visualization Specifications

This section presents two examples of how the HENSON framework can be applied to specify GA visualizations. The first example explains the process involved in specifying the design of the commonly used fitness versus time graph. This illustrates how the Henson framework can structure the analysis of the user’s queries in order to identify the relevant information and interesting events from the GAs execution. This enables the definition of the HENSON *players* and *history events* required for the visualization. From the definition of these items, the user can select an appropriate view from the supplied GA specific view hierarchy and identify the necessary mappings. The second example illustrates how versatile the HENSON framework is by presenting the specification of the VIS GA visualization tool recently developed by Annie Wu (see Section 4.2).

6.4.1 Fitness vs Time Graph

This subsection presents the HENSON specification for a fitness versus time (i.e. generation number) graph. This illustrative example shows the best, average and worst fitness ratings from each population in a GA’s run (see Figure 6.4 on page 171). Table 6.8 summarizes the HENSON definition for

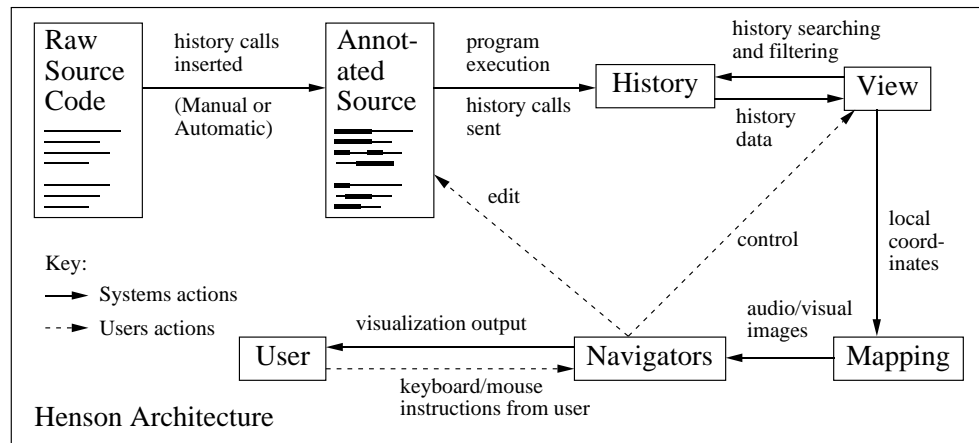


Figure 6.3: The architecture of the HENSON GA visualization framework. The four main modules; History, View, Mapping and Navigators, are shown in rectangular boxes, as in the VIZ architecture diagram. However, HENSON links the navigator module to the visualization source module in order to enable the user’s editing of the GA’s parameters or components.

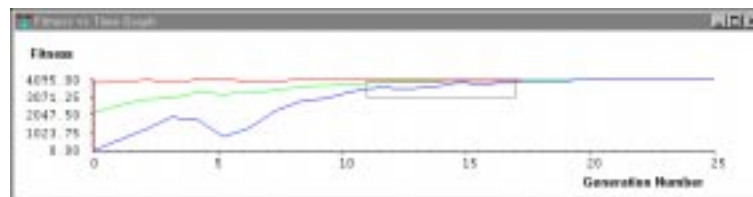


Figure 6.4: An example GA visualization, a fitness versus time graph showing the minimum (blue line), average (green line) and maximum (red line) fitness ratings in each population for a 29 generation GA run (0 to 28).

this visualization. The ecosystem’s statistics slot is used here as the GA’s history module, and the population-statistics slot in the ecosystem’s statistics class is used as the main player. The state of the population-statistics object after each regenerate event is stored in the ecosystem’s statistics slot. The recorded history is then accessed by three fitness-mappings to produce a 2D-fitness-v-time-graph view. Minimum, average and maximum fitness line mappings map the values of the population-statistics’ min-score, avg-score and max-score.

6.4.2 Vis - GA Visualization Tool

As previously noted in Section 4.2, VIS supports the visualization of a GA’s chromosomes from a stored dataset of a GAs execution. The user can view the best individuals from the GA’s entire run in the “run window” (see Figure 4.13 on page 94), all the individuals in the population for a

Table 6.8: The HENSON definition of the example fitness versus time graph given in Figure 6.4.

ENTITY	NAME	SUPERIORS	SLOTS
<i>history</i>	(statistics (population ecosys- tem))		
<i>player</i>	population-statistics		
<i>event</i>	regenerate		
<i>view</i>	2D-fitness-v-time- graph	2D-line-graph	
<i>mapping</i>	fitness-mapping		(view 2D-fitness-v-time-graph)
<i>mapping</i>	minimum-fitness-line ——	fitness-mapping	(previous (min-score population-statistics)) (next (min-score population-statistics)) (view 2D-fitness-v-time-graph)
<i>mapping</i>	average-fitness-line ——	fitness-mapping	(previous (avg-score population-statistics)) (next (avg-score population-statistics)) (view 2D-fitness-v-time-graph)
<i>mapping</i>	maximum-fitness-line ——	fitness-mapping	(previous (max-score population-statistics)) (next (max-score population-statistics)) (view 2D-fitness-v-time-graph)

specific generation in the “population window” (see Figure 4.14 on page 95), or a specific individual in a specific generation in the “individual window” (see Figure 4.15 on page 96). VIS allows the user to select any one of five representations for their chromosomes, these five options are referred to as; “text,” “zebra,” “neapolitan,” “colour coded,” or “gene location” representations.

The HENSON specification for VIS is summarized in Tables 6.9, 6.10 and 6.11. The main players in the visualization are the GA’s chromosomes. The players that need to be recorded in the history module are the organisms’ chromosomes, their fitness ratings, parents, crossover points, and mutated bits. These players’ status should be recorded in the history module during the reproduction process. The three views present the chromosomes in a common format and should be defined as subclasses of the HENSON “chromosome icons” view (as shown in Table 6.9). The navigators illustrated in the

Table 6.9: The HENSON definition of the history, players, events and views used in the VIS GA visualization tool.

ENTITY	NAME	SUPERIORS	SLOTS
<i>history</i>	(statistics (population ecosystem))		
<i>player</i>	population-statistics		organism parent-data crossover-data mutation-data
<i>event</i>	regenerate		
<i>view</i>	run-window	chromosome-icons	
<i>view</i>	population-window	chromosome-icons	
<i>view</i>	individual-window	chromosome-icons	

given screen views include a system menu, view-specific buttons, and a text box for identifying lines or individuals to be displayed (see Table 6.10 for the HENSON specification). The mappings used to display the chromosomes follow the five representations noted above (text, zebra, neapolitan, colour coded and gene location, see Table 6.11).

6.5 Summary

This chapter has introduced “HENSON” a supportive framework for the development of interactive GA visualizations. The contribution of this framework is that it provides a structured support environment which includes a set of GA-specific players and a GA-specific view hierarchy that supports the construction of the GA visualizations reviewed in Section 4.1.

New visualizations can be introduced by identifying the players and events of interest from the set listed above (Table 6.5), selecting the appropriate views from the HENSON view hierarchy (Figure 6.3), and establishing a set of mappings to display the values of the players as graphical objects in the chosen views. In addition to applying the GA specific components provided in HENSON to produce new types of visualization, additional user-defined components can be introduced by the user at any time. In this way the user is not limited to the visualizations provided by the system, rather the modular components in HENSON facilitate the extension of the framework. For example, in order to

produce a new view the user can create a subclass of the most appropriate existing view in the view hierarchy, and by adding the necessary specializations, create a new view.

Table 6.10: The HENSON definition of the *navigators* used in the VIS GA visualization tool.

NAME	SLOTS
(system-menu (view run-window))	window-menu which-menu view-menu view-menu
(system-menu (view population-window))	window-menu display-menu population-menu view-menu
(system-menu (view individual-window))	window-menu display-menu view-menu format-menu
run-window-control-panel	(view run-window) line-up-button line-down-button page-up-button page-down-button refresh-button goto-line-text
population-window-control-panel	(view population-window) line-up-button line-down-button page-up-button page-down-button refresh-button goto-line-text
individual-window-control-panel	(view individual-window) previous-individual-button next-individual-button undo-button goto-individual-text

Table 6.11: The HENSON definition of the *mappings* used in the Vis GA visualization tool.

NAME	SLOTS
text-chromosome	(chromosome (genotype organism) (view run-window) (view population-window) (view individual-window)
zebra-chromosome	(binary-chromosome (genotype organism) (view run-window) (view population-window) (view individual-window)
neapolitan-chromosome	(binary-chromosome (genotype organism) (view run-window) (view population-window) (view individual-window)
colour-coded-chromosome	(chromosome (genotype organism) (view run-window) (view population-window) (view individual-window)
gene-location-chromosome	(initial-population) (chromosome (genotype organism) (view run-window) (view population-window) (view individual-window)

Chapter 7

GONZO: A Search Space

Visualization Tool

This chapter presents “GONZO” a GA visualization tool designed to support peoples understanding of their GA’s search behaviour. The design of GONZO is described in Section 7.1. The design features, intended to fulfill the visualization requirements established in the user study (Section 3.3), are discussed in Subsection 7.1.1. The design specification of GONZO, using the HENSON framework introduced in the previous chapter, is presented in Subsection 7.1.2. Section 7.2 describes the resulting implementation of GONZO, and Section 7.3 explains how GONZO can be applied to produce offline and online visualizations. Section 7.4 describes a series of example problems and illustrates how GONZO can be used to explore a GA’s search behaviour. Section 7.5 describes a menu based graphical user interface which can be used as a front end to GONZO. Section 7.6 explains how GONZO can be used in practice - the use of a “GA Examples” menu to illustrate the example problems using the default visualizations is described along with the introduction of new visualizations and new GAs. Section 7.7 concludes this chapter with a summary of the visualization attributes of GONZO.

7.1 Design

This section presents the design features and specification of GONZO. GONZO is designed to fulfill the set of users’ questions concluded from the GA user study (Section 3.3), specifically:

- How diverse/converged are the chromosomes in the population?
- Are there clusters of chromosomes forming during the GA's run?
- How does the local structure of the chromosomes affect the chromosomes' fitness ratings?

As pointed out in the design rationale chapter, search space visualizations can be used to show the population's sampling of the search space. In doing so the diversity, convergence, and the formation of clusters in the population can be observed. Furthermore, by using a structurally based representation, such as the extensive repartitions technique described in Section 5.3.3, the local structure of the chromosomes can be derived from their location in the search space representation and the relationship between the chromosomes' fitness ratings and local structure can be explored.

In terms of an SV taxonomy [Price et al., 1993], SVs can be designed to support the user's understanding of either the program or algorithm. Program visualizations support the user's understanding of the program's code and data values, where as algorithm visualizations support the user's understanding of the algorithm's instructions and generic data structures. Visualizing the algorithm's instructions, such as "select chromosomes for reproduction," "crossover two parents to form two children," or "mutate chromosome," presents the actions involved in running a GA. Although this is at a high level of abstraction in terms of the program, it is at a fine-grained level in terms of the algorithm's behaviour.

As indicated by the questionnaire responses (see Section B.3, Question 8) this may be useful as an educational or debugging aid but should be used selectively as it presents so much fine-grained detail of the GAs execution. GONZO is an algorithm visualization system primarily concerned with illustrating the algorithm's "high level" data structures. Visualizing the work of the algorithm presents the user with a more direct view of the GA's search behaviour than visualizations of the code or the "low-level" data values.

By applying the search space matrix technique described in Section 5.3.3 GONZO displays a search space visualization of the GA's population data (i.e. the chromosomes and fitness ratings). In order to show the GA's evolutionary search behaviour, the user must have some way of identifying the temporal context of the search space visualization. This is achieved through the use of an augmented fitness versus time graph which displays a coarse-grained view of the GA's history and highlights the

region currently illustrated in the search space visualization. In addition to seeing each chromosome as a unique point in the search space visualization, the individual points can also be selected and the corresponding chromosome value and its fitness rating can be displayed in a fine-grained chromosome view.

These three visualizations provide three coupled views of the GA's population. The complete GA run is shown in the fitness versus time graph, the individual chromosomes in the generation range and fitness rating range (highlighted by a grey rectangle in the fitness versus time graph) are shown as points in the search space visualization, which can be selected by the user and the corresponding chromosome details (including the chromosome value and fitness rating) will be displayed in the fine-grained chromosome view.

With reference to the conclusions of the GA user study (Section 3.3), GONZO is an interactive tool that aims to support the user's interpretation of the algorithm's behaviour (*supportive*). The user can navigate backwards and forwards through each generation of the algorithm's execution and pause, edit and restart the algorithm (*interactive*). GONZO can be applied directly with its current functionality or extended, through the application of the HENSON framework, to provide additional visualization support (*usable* and *expressive*).

7.1.1 Interface Design

This subsection describes the individual views and navigators available in GONZO. Figure 7.1 shows an example screen image taken from GONZO containing a coarse-grained fitness versus time graph (bottom right), a medium-grained search space visualization (top right), a fine-grained chromosome view (bottom left), a movie player control panel (top left), a generation and fitness range selector (second left), and a schema highlight selector (third left). The design features of each component are described in the remainder of this subsection.

Augmented Fitness versus Time Graph

The "augmented fitness versus time graph" shows the results of the GA's run. The values of the best, worst and averaged fitness ratings from each generation are plotted on a three line graph (see Figure 7.2). In GONZO an additional rectangle is plotted on top of the fitness versus time graph, the

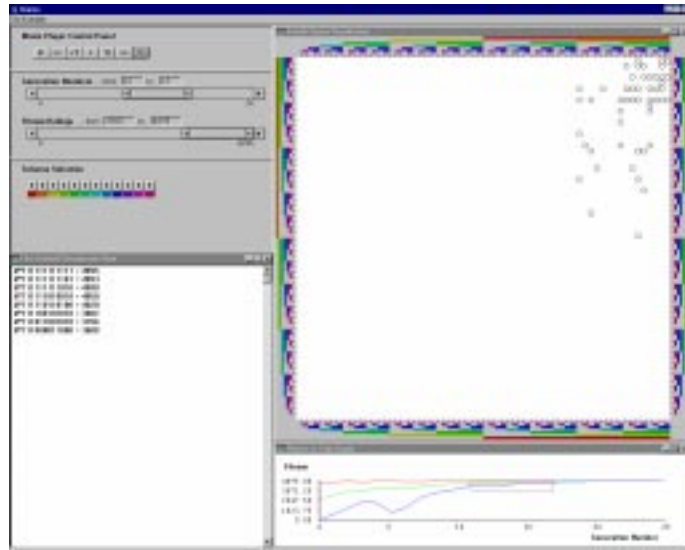


Figure 7.1: An example screen image taken from GONZO. This example includes three views; a coarse-grained fitness versus time graph (bottom right), a medium-grained search space visualization (top right) and a fine-grained chromosome view (bottom left), and three navigators; a movie control panel (top left), generation and fitness range selector (second left), and a schema highlight selector (third left). The search space visualization currently illustrates all the chromosomes considered by the GA between generations 11 and 17 with fitness ratings between 3050 and 4095. The GA is attempting to solve the 12 bit maximum integer problem. This problem seeks out binary chromosomes with high integer values.

width and height of the rectangle indicate the range of generations and fitness ratings currently being displayed in the search space visualization.

Search Space Visualization

The “search space visualization” shows the GA’s chromosomes as a set of points in a 2D representation of the GA’s search space (see Figure 7.3). This visualization contains two parts; the “search space view” and the “schema legend.” The *search space view* shown in the centre of the search space visualization translates each chromosome to a coordinate and displays it as a point image. Different image mappings can be used to identify each chromosome’s fitness and/or frequency in the population. The *schema legend* shown around the outside edge of the search space visualization is used to identify regions of the search space view. The “Schema Highlighting Dialog” (see below) can be used to identify schemata of interest to the user. The regions of the search space view that contain elements of the

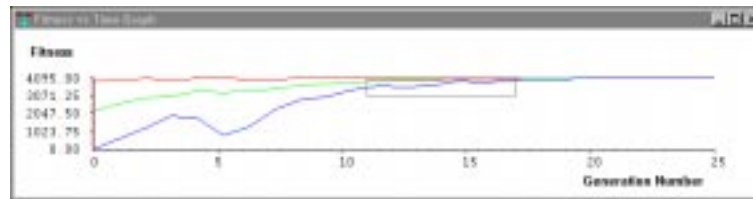


Figure 7.2: An example of the fitness versus time graph available in GONZO. The top red line trace indicates the best fitness rating in each population, the bottom blue line trace indicates the worst fitness rating in each population, and the middle green line trace indicates the average value of all the chromosomes' fitness ratings. A superimposed grey rectangle indicates the region of the GA's run that is currently being displayed in the search space visualization (i.e. the chromosomes considered between generation 11 and generation 17 with fitness ratings between 3050 and 4095).

user defined schema will then be highlighted in the schema legend. The example given in Figure 7.3 highlights the regions of the search space with allele “1” at every locus, each coloured ribbon in the schema legend indicates a different locus (see Figure 7.7 for the associated schema highlighting dialog).

Three different image mappings are supported in GONZO's search space view: size, value and colour. Both size and value are dissociative image variables that bias the user's attention toward the larger or more contrasting (i.e. darker) chromosome images. This can be useful to draw the user's attention toward the fitter or more common chromosomes in the population. Size and value also support the user's perception of order. Although size supports the perception of quantities, within a 2D representation the amount of screen space available for each chromosome limits the range of quantities that can be displayed. Although colour does not support a natural ordering it does support the formation of associations and selections, the user can perceive differently coloured items and group them together to form families. This can be useful in terms of identifying sections of the GA's fitness surface. For example, if the colour spectrum blue through to red is used to show fitness ratings from low to high, the user can easily identify the regions of the search space that have a low fitness (blue), average fitness (green) and high fitness (red). According to [Bertin, 1983] up to seven different colours can be used to support visual selection (see Section 5.2).

The user's choice of image mapping should be guided by their visualization requirements; identifying convergence and clustering requires the user to note the regions of the search space with good

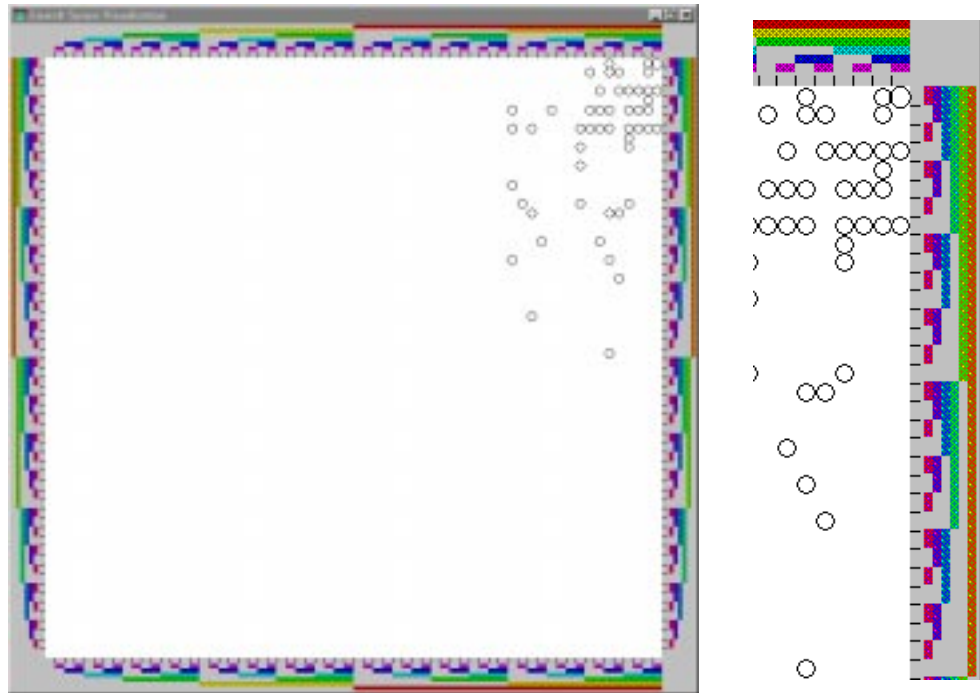


Figure 7.3: An example of a search space visualization. The search space visualization is shown here on the left with an enlarged section of its schema legend shown on the right, the string highlighted here is 111111111111. Each chromosome is indicated as a point, at each point a circle is drawn, where the size of the circle indicates the chromosome's fitness. The circles range from small circles for low fitness values to large circles for a high fitness values.

fitness ratings, and dissociative image variables such as size and value are useful for this. Identifying the relationship between the chromosomes' local structure and fitness however, requires the user to draw associations between the chromosomes' local structure in different regions of the search space and their fitness ratings (low or high), colour is the most effective image mapping for achieving this requirement.

Fine-Grained Chromosome View

The "fine-grained chromosome view" presents the values of selected chromosomes and their fitness ratings (see Figure 7.4). This view is coupled to the search space visualization and supports the user's further investigation of the chromosomes in the search space.

When this view is displayed the user can select chromosomes by clicking in the search space view. Providing the resolution of the matrix is less than or equal to the resolution of the screen display area,

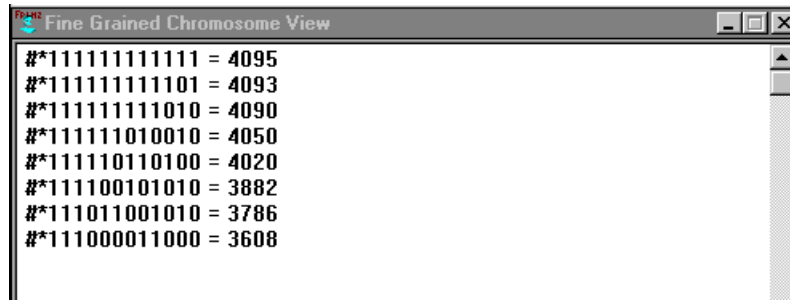


Figure 7.4: An example of a fine-grained chromosome view displaying the chromosome value and fitness rating of eight selected chromosomes.

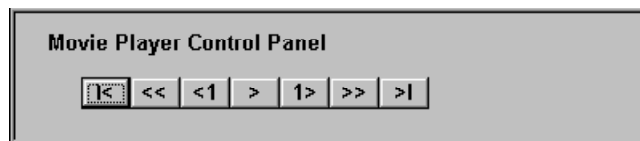


Figure 7.5: The movie player control panel used in GONZO to navigate the GA's execution.

the coordinate of the cursor when the mouse button is released is translated back into a chromosome genotype which is displayed in the fine-grained chromosome view along with the chromosome's fitness rating.

If the resolution of the search space matrix is greater than the resolution of the screen display area then as much of the chromosome as possible is identified and displayed in the fine-grained chromosome view. Any alleles in the chromosome which cannot be uniquely identified are given the first value in the coding alphabet.

Movie Player Control Panel

The “movie player control panel” enables the user to navigate through the GA's execution a generation at a time. The user can either go back to the start of the run (“| <”), step back a number of generations (a default of ten, “<<”), step back one generation (“< 1”), play or pause the run like a movie (“>” or “||”), step forward one generation (“1 >”), step forward a number of generations (“>>”), or go forward to the last generation (“> |”).

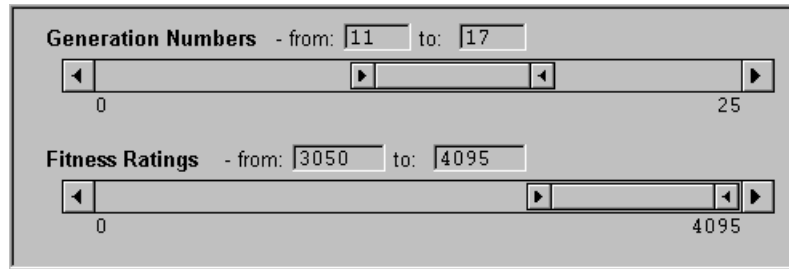


Figure 7.6: The alphaslider range selector used in GONZO to define a range of generations and fitness ratings to be displayed in the search space view.

Generation and Fitness Range Selector

As well as navigating the GA’s execution via the movie player control player, the user can also select a range of generations, or a range of fitness ratings, to be displayed using the “generation and fitness range selector.” This navigator includes two alphasliders for defining the range of generation numbers and fitness ratings to be displayed in the search space view and highlighted in the augmented fitness versus time graph.

These alphasliders can be manipulated in seven ways, the exterior arrow buttons step the current range either one position to the left or one position to the right, pressing the mouse button in the region between the central range bar and exterior buttons steps the range to the left or right by ten percent of the total range, dragging either the left or right arrow buttons on the central range bar changes the defined range, and dragging the middle section of the central range bar moves the defined range. As well as directly manipulating the alphaslider the user can manually edit the text fields in the “from” and “to” text boxes shown directly above each alphaslider.

In addition to updating the search space visualization to show the chromosomes contained in the defined range, the fitness versus time graph is updated to illustrate the defined range as a rectangular bounding box. The bounding rectangle in the fitness versus time graph links the fitness graph with the search space visualization, the movie player control panel, and the generation and fitness range selector.



Figure 7.7: The schema highlighting dialog used in GONZO. The location of the chromosomes containing the schema identified by the button labels are highlighted using the colour coded ribbons in the legend of the search space matrix. Clicking on each button makes the label change to the next allele in the GA's coding alphabet.

Schema Highlighting Dialog

The “schema highlighting dialog” enables the user to highlight sections of the search space view that contain specific alleles. The regions of the schema legend that can be highlighted are constrained to those regions of the search space view that can be effectively displayed within the screen resolution available. The button labels on the schema highlighting dialog define the allele to be highlighted in the search space visualization. Pressing each button makes the label change to the next symbol in the coding alphabet until the last value then a wild card symbol (“*”) is shown to indicate that no alleles are being highlighted for that locus and the sequence starts again. Hence, the schema highlight selector used for binary representations displays the sequence *, 0, 1, *, 0, . . . etc.

7.1.2 HENSON Specification

This subsection presents the design specification of GONZO, as described in the previous subsection, using the HENSON framework presented in Chapter 6. Table 7.1 defines the “players” and “events,” Table 7.2 defines the “views,” Table 7.3 defines the “mappings” and Table 7.4 defines the “navigators.”

Players and Events

The main *player* in GONZO, i.e. the main item of interest, is the GA's population statistics and how they change during the GA's run (see Table 7.1). The population-statistics component of GONZO has four slots of interest to the views used here; the min-score, the max-score, the avg-score and the population. The minimum, maximum and average scores (i.e. fitness ratings) are used to produce the fitness versus time graph and the population data, including all the organisms' chromosome values and fitness ratings, are used to produce the search space visualization.

Table 7.1: The HENSON definition of the *players* used and *events* recorded in GONZO.

MODULE	NAME	SLOTS	DESCRIPTION
<i>Players</i>	population-statistics	population sum-score avg-score max-score min-score max-organism min-organism sum-normalized-score avg-normalized-score	The main (default) player used in Gonzo to record statistics regarding each population.
	population		The GA's population.
	avg-score		The average fitness rating in a population.
	max-score		The maximum fitness rating in a population.
	min-score		The minimum fitness rating in a population.
<i>Event</i>	evaluate (population)		After each population evaluation record the GA's population statistics.

Table 7.2: The HENSON definition of the *views* available in GONZO.

NAME	SUPERIORS	SLOTS
augmented- fitness-v-time- graph	2D-fitness-v-time-graph	(elements-of-plot (min-score population-statistics) (avg-score population-statistics) (max-score population-statistics)) (element-coord-fun fitness-line-mapping) (highlight-region generation-range fitness-range)
search-space- visualization		search-space-matrix schema-legend
search-space- matrix	2D-point-plot	(elements-of-plot organisms) (element-coord-fun search-space-chromosome-mapping) (display-focus generation-range fitness-range)
schema-legend	2D-point-plot	(elements-of-plot highlight-schema) (element-coord-fun schema-mapping)
fine-grained- chromosome- view	formatted-text	(elements-of-plot chromosome fitness-rating)








In order to follow the progress of the GA, generation by generation, the player information must be recorded in the History module every generation; this is done after each population is evaluated by the `evaluate (population) event`.

Views

There are three *views* in GONZO: the fitness versus time graph, the search space visualization, and the fine-grained chromosome view (see Table 7.2). The fitness versus time graph is a specialized version of the 2D line graph that includes a rectangle highlighting the current range of generation numbers and fitness ratings identified by the generation and fitness range selector.

The search space visualization is made up of two sub-views - the search space matrix and the

Table 7.3: The HENSON definition of the *mappings* used in GONZO.

NAME	SLOTS
min-score-line 	(entity (min-score population-statistics)) (view fitness-versus-time-graph)
avg-score-line 	(entity (avg-score population-statistics)) (view fitness-versus-time-graph)
max-score-line 	(entity (max-score population-statistics)) (view fitness-versus-time-graph)
generation-fitness-range-box 	(entity generation-range fitness-range) (view fitness-versus-time-graph)
chromosome-icon 	(entity organism) (view search-space-matrix)
schema-ribbon  	(entity highlight-schema) (view schema-legend)
organism-details 100101011101 0.85	(entity organism) (view fine-grained-chromosome-view)

schema legend, both of which are specialist forms of a 2D point plot. The search space matrix has a white background with a calibration scale around the edge of the view and it contains the mappings that link the chromosomes to the point images and the navigator used to identify chromosomes of interest. The schema legend has a grey background and contains the mapping that links the schema identified in the schema highlighting dialog to the coloured ribbons drawn in the legend.

Finally, the fine-grained chromosome view shows the chromosomes and fitness ratings of selected chromosomes from the search space visualization. This is a specialized form of a text view that displays the chromosome value and score of the organisms identified by the chromosome navigator in the search space visualization.

Mappings

There are seven *mappings* required to produce GONZO (see Table 7.3):

- three line mappings to produce the fitness versus time graph, i.e. the min-score-line, avg-score-line and max-score-line;
- one empty rectangle mapping on the fitness versus time graph to show the current range of generation numbers and fitness ratings being displayed, i.e. the generation-fitness-range-box;
- a filled rectangle to indicate each chromosome in the search space matrix, i.e the chromosome-icon;
- a filled rectangle mapping to highlight the value of the schema selection dialog in the schema legend of the search space visualization, i.e. the schema-ribbon;
- finally, an organism details mapping is used to display a selected organism's chromosome value and fitness rating in the fine-grained chromosome view.

Navigators

Four *navigators* are used in GONZO: the movie player control panel, the generation and fitness range selector, the schema highlight selector, and the search space chromosome navigator (see Table 7.4):

- the movie player control panel sets the value of the views' current generation range and refreshes the appearance of any associated views and navigators, i.e. the generation and fitness range selector, the search space matrix, and the fitness versus time graph;
- the generation and fitness range selector sets the values of the views' current generation range and current fitness range and refreshes the fitness versus time graph and search space visualization;
- the schema highlight selector sets the value of the schema legend's highlight schema and refreshes the schema legend view;
- finally, the search space chromosome selector sets the value of the fine-grained chromosome view to include the chromosome details identified by the cursor's position in the search space view, and refreshes the fine-grained chromosome view to include the added information.

Table 7.4: The HENSON definition of the *navigators* available in GONZO.

NAME	SLOTS
movie-player-control-panel	(set-value generation-range) (update movie-player-control-panel) (update generation-and-fitness-range-selector) (update search-space-matrix) (update fitness-versus-time-graph)
generation-and-fitness-range-selector	(set-value generation-range) (set-value fitness-range) (update generation-and-fitness-range-selector) (update search-space-matrix) (update fitness-versus-time-graph)
schema-highlight-selector	(set-value highlight-schema) (update schema-selection-dialog) (update schema-legend)
search-space-chromosome-selector	(set-value fine-grained-chromosome-view) (update fine-grained-chromosome-view)

In addition to the changes that these navigators make to their respective views, they also update their own display to reflect their current value. The play/pause button in the movie player control panel toggles between play (“>”) and pause (“||”) to reflect the current state of the player, the generation and fitness range selectors show the current values of the ranges they define, and the schema selection dialog identifies the current schema shown in the schema legend.

7.2 Implementation

GONZO was implemented in Allegro Common Lisp using CLOS¹ on an IBM compatible PC, running Windows NT. Unfortunately within the bounds of this project there was insufficient time available

¹The Common Lisp Object System (“CLOS”) is an object oriented version of Lisp.

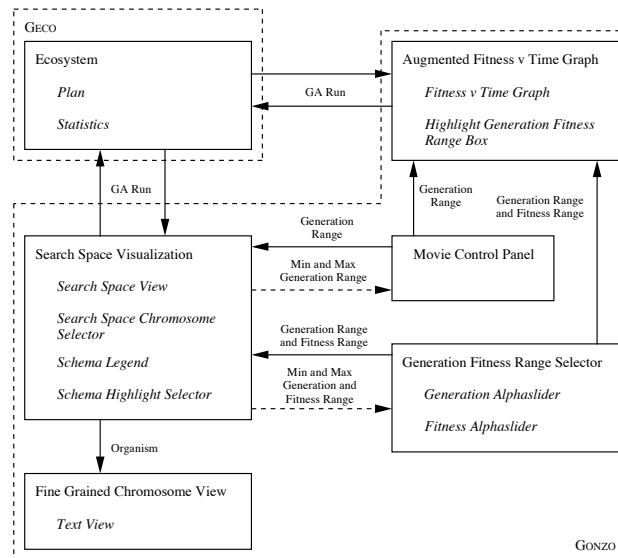


Figure 7.8: The architecture of GONZO. Here the distinction between the GECO GA prototyping environment and GONZO visualization tool is made explicit along with the content and direction of communication made between each module. Dashed lines are used to distinguish the information required to initialize GONZO.

to build the complete HENSON framework and so only those features necessary to illustrate the functionality of the framework and to build GONZO were fully implemented: specifically the 2D fitness graph, and 2D point plot views, the search space matrix mapping, and the movie player control panel and alphaslider navigators. These components were used to produce GONZO.

A generic GA prototyping framework called “GECO” was adopted as a GA environment for GONZO. “GECO” is an abbreviation of Genetic Evolution through Combination of Objects and is a CLOS-based framework for prototyping GAs [Williams, 1993]. The distinction between GECO and GONZO is illustrated in the architecture diagram shown in Figure 7.8. GECO is used here as a stand-alone GA prototyping environment. GONZO can be used, either online or offline, with GECO to illustrate the execution of a GA. This section describes the implementation of GONZO, explaining the definition and operation of each component.

History Data - GA Run

In GECO the execution of the algorithm is recorded by default in the population’s `statistics` slot as a set (i.e. vector) of `population-statistics` class instances (see Table 7.1). The `statistics` slot of the GA’s population is used in GONZO as the visualization’s `history` module. The `avg-score`,

`max-score` and `min-score` players are used to draw the fitness versus time graph, and the `population` player is used to draw the search space visualization.

The GECO `compute-statistics` method called within the `evolve` method records this information by default. If additional information is required in future visualizations then the `compute-statistics` method can be extended to record the necessary additional information. The GA's execution can either be held in memory or stored in a data file. For the examples presented in Section 7.4 the execution history is stored in memory.

Search Space Visualization

The search space visualization component of GONZO includes the search space matrix, the schema legend, the search space chromosome selector and the schema highlight selector. The search space matrix, schema legend and search space chromosome selector are created by the `create-search-space-visualization` command.

```
(create-search-space-visualization
  name dataset chromosome-mapping-technique parent-dialog exterior-box
  &optional coordinate-mapping-technique list-of-views projection-locus-order)
```

The argument *name* is used to identify the view, the *dataset* identifies the History module being used, the *chromosome-mapping-technique* identifies the mapping being applied (in this case the search space matrix mapping although any mapping or look-up function could be used), the *parent-dialog* identifies the dialog in which the search space visualization will appear and the *exterior-box* identifies the box containing the view using the local coordinates of the parent-dialog.

The *coordinate-mapping-technique* identifies the coordinate to chromosome mapping method used in the fine-grained chromosome view. The *list-of-views* argument is used to identify the fine-grained chromosome view. Finally, the *projection-locus-order* identifies a list of locus orderings for the search space matrix mapping. The default projection-locus-order is from left to right, i.e. (0 1 2 3 4 5 6 7 8 9 10 11 12), although any ordering could be specified, e.g. right to left (12 11 10 9 8 7 6 5 4 3 2 1 0), or half and half on each axis of the matrix (0 6 1 7 2 8 3 9 4 10 5 11 12).

Within GONZO any changes to the current generation range or current fitness range cause the search space view (and augmented fitness versus time graph, see below) to be refreshed. In order to redraw the search space view, GONZO first compares the old set of organisms with the new set of organisms. Those organisms that no longer need to be shown are then drawn over using the background colour of the display window, and those organisms that do need to be shown, i.e. those within the new range, are drawn using the specified mapping. Any changes made to the alleles in the highlight schema cause the schema legend to be updated, in this case only the individual sections of the schema legend that relate to the locus of the changed allele are erased and redrawn.

Selecting points within the search space view invokes the search space chromosome selector which takes the local coordinate position of the cursor when the mouse button is released and translates the coordinate back into a chromosome. The chromosome is then used to create an organism which is passed to the fine-grained chromosome view and its chromosome value and evaluated fitness rating is displayed.

Schema Highlight Selector

```
(create-schema-highlight-selector
  name list-of-views parent-dialog exterior-box)
```

Even though within the architecture of GONZO the schema highlight selector is a part of the search space visualization (see Figure 7.8), it is created independently of the search space visualization. There are two reasons for this: the location of the schema highlight selector is different to that of the search space visualization, and a single schema highlight selector could be used with multiple search space visualizations. The schema value of the selector initially defaults to a string of wild card symbols, i.e. nothing is highlighted in the schema legend of the search space visualization(s). The *list-of-views* variable identifies each of the views to be updated when the value of the schema selection dialog is changed, and the *exterior-box* identifies the position and size of the schema highlight selector in the *parent-dialog*. The number of schema buttons and their range of values is determined by the population data associated with the first view given in the *list-of-views*. When changes are made to the highlight schema the schema legend of the search space visualization is updated as described

above.

Augmented Fitness Versus Time Graph

```
(create-fitness-versus-time-graph  
  name dataset parent-dialog exterior-box)
```

The fitness versus time graph uses the same *dataset* as the search space visualization, the *parent-dialog* and *exterior-box* identify the parent window and the position and size of the fitness versus time graph.

Within GONZO any changes made to the total generation range and total fitness range cause the *entire* fitness versus time graph to be redrawn to include the complete range of generation numbers and fitness ratings. Changes to the current generation range and fitness range cause the *contents* of the fitness versus time graph to be redrawn. Redrawing the contents of the fitness versus time graph involves clearing everything except the axes and labels of the graph and drawing the average, maximum and minimum fitness lines, and the rectangular box highlighting the current generation and fitness range.

Fine-Grained Chromosome View

```
(create-fine-grained-chromosome-view  
  name parent-dialog exterior-box)
```

The last view is the fine-grained chromosome view; this is not linked directly to the dataset, it simply displays the data that is passed to it by the search space chromosome selector. The search space chromosome selector is a navigator included in the search space visualization. When creating a fine-grained chromosome view the *name*, *parent-dialog* and *exterior-box* are the only arguments used.

Movie Player Control Panel

```
(create-movie-player
```

```
name list-of-labels list-of-functions list-of-views parent-dialog exterior-box)
```

The movie player takes a list of button labels and function names and creates a button for each label and function pair. Each button's title is set to the label value and the button's function is set to the function name as specified in the *list-of-labels* and *list-of-functions*, respectively. The list of views are updated every time the buttons are used. The location and size of the movie player are identified by the *parent-dialog* and *exterior-box* arguments. When called the individual functions change the value of the current range of generations displayed by each view identified in the *list-of-views* list.

A set of seven functions are available in GONZO; *start* sets the views' *generation-range* to 0, *rewind* sets the views' *generation-range* back 10 generations, *back1* sets the views' *generation-range* back 1, *play-pause* either periodically steps forward a single generation per second or pauses, *forward1* sets the views' *generation-range* forward 1, *fforward* steps the views' *generation-range* forward 10, and *end* sets the views' *generation-range* to the last generation in each view. Changing the value of the generation range also updates the augmented fitness versus time graph and the search space visualization as described above.

Generation and Fitness Range Selector

```
(create-generation-fitness-selector
```

```
name list-of-views parent-dialog exterior-box)
```

```
(create-alpharanger
```

```
name title-string list-start-and-end-values list-start-and-end-range
```

```
set-value-function exterior-box)
```

The generation and fitness range selector has a specific creation function that calls the *create-alpharanger* function twice to produce a generation number range selector and a fitness rating range selector. The minimum and maximum values of the generation numbers and fitness ratings associated with each of the views included in the *list-of-views* are used to define the *list-start-and-end-range* of the two range selectors. The default initial value of the

list-start-and-end-values argument for the generation range selector is to start and end at generation 0. The default initial range value of the *list-start-and-end-values* argument of the fitness rating range selector is a list of the minimum and maximum fitness ratings found in the dataset for each associated view. Like the movie player control panel, changing the value of the generation range, or the fitness range, also causes the augmented fitness versus time graph and search space visualization to be updated.

7.3 Application

This section explains how the above GONZO implementation can be applied to produce both offline and online visualizations. The actual Lisp code used to produce the examples presented in this chapter is included in Appendix D.

7.3.1 Offline Visualization

To produce the offline visualization shown in Figure 7.1 a `test` function was defined first to run the GA and then to create the visualizations. The GA is run by calling the GECO `test-plan` function which takes three arguments; a *name*, the *number-of-runs* and a *GA-plan*. The *name* argument is then used to access the results of the GA, and this is used in the `create-visualizations` method to produce the offline visualizations. The `create-visualizations` method creates the `*visualization-dialog*` using the standard common-graphics `open-dialog` command, and this is used as the parent-dialog for all of the GONZO components.

The fine-grained chromosome view, and search space visualization are both produced in a similar manner to the `fitness-versus-time-graph`, which is created as follows:

```
(create-fitness-versus-time-graph
  'fitness-graph-0 ;; name
  ga-run ;; dataset
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 400 0 1278 204)) ;; exterior-box
```

The same `ga-run` dataset and `*visualization-dialog*` parent-dialog are used in all of the views, while the view names and exterior box dimensions are specified individually. The three navigators are also created in a similar way to one another, for example the `generation-fitness-selector` is created by the command:

```
(create-generation-fitness-selector
  'view-range-window ;; name
  (list fitness-graph-0 scatterplot-view-0) ;; list-of-views
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 0 80 400 240)) ;; exterior-box
```

7.3.2 Online Visualization

The previous example has shown how GONZO can be applied to produce an offline visualization of a GA's recorded history, this section describes how the same views can be produced for online visualizations. The difference between offline and online visualizations is that offline visualizations are produced after the algorithm's execution and online visualizations are produced after the initial generation has been evaluated and they are then updated after each consecutive generation.

To do this in GECO the `create-visualizations` function is called from within the GECO `EVOLVE` method. Updating the visualization to follow the evolution of the GA is done by incrementing the `current-generation-range`, the `total-generation-range` and (if necessary) the `current-fitness-range` and `total-fitness-range` of the views contained in the `*visualization-dialog*`. The annotated version of the GECO `EVOLVE` method is included in Appendix D.

Annotating the GECO `EVOLVE` method to create and update the visualizations is the only essential difference between producing offline and online views in GONZO. Updating the `total-fitness-range` variable for each view as indicated in Appendix D has the effect of redrawing the fitness graph with the expanded total range as well as updating the values of any associated navigators. During the course

of the GA's execution, particularly in the initial generations when this range changes frequently, the fitness versus time graph can appear to flicker. This flickering can be avoided by identifying an expected total fitness range when the visualizations are first created in the `create-visualizations` function.

7.3.3 Interactive Command Line Control

Finally, because Lisp is an interpreted, rather than a compiled language, the visualization commands available in GONZO, as well as the GA commands available in GECO, can be applied via the command line either during the GA's run (online) or whilst an offline visualization is being displayed. In this way, the user is un-restricted in the control they have over both the GA and the GA visualization. A similar degree of freedom is available in the SAMBA visualization tool [Stasko et al., 1993] (see Section 4.1.4).

7.4 Example Problem Visualizations

This section illustrates some of the applications in which GONZO has been used to explore the search behaviour of GAs. The problems investigated here are the maximum integer problem used previously as an example, the De Jong F1 test problem [De Jong, 1975] (as reviewed in [Goldberg, 1989]), and the Royal Road function [Mitchell et al., 1991].

7.4.1 The Maximum Integer Problem

The relatively simple maximum integer problem is used as a common example used for illustrating the execution of the GA in an educational context. For this problem the GA attempts to maximize the integer value of the chromosomes in the population. This is an easy problem for students to follow as they are well aware of the concept of binary to integer number translation and can therefore understand the link between the organisms' chromosome values and fitness ratings.

As the GA progresses the initially random distribution of points migrates towards the top right hand corner of the search space, as shown in Figure 7.9. In the search space visualizations shown above the projection ordering used places the even loci along the horizontal axes and odd loci along

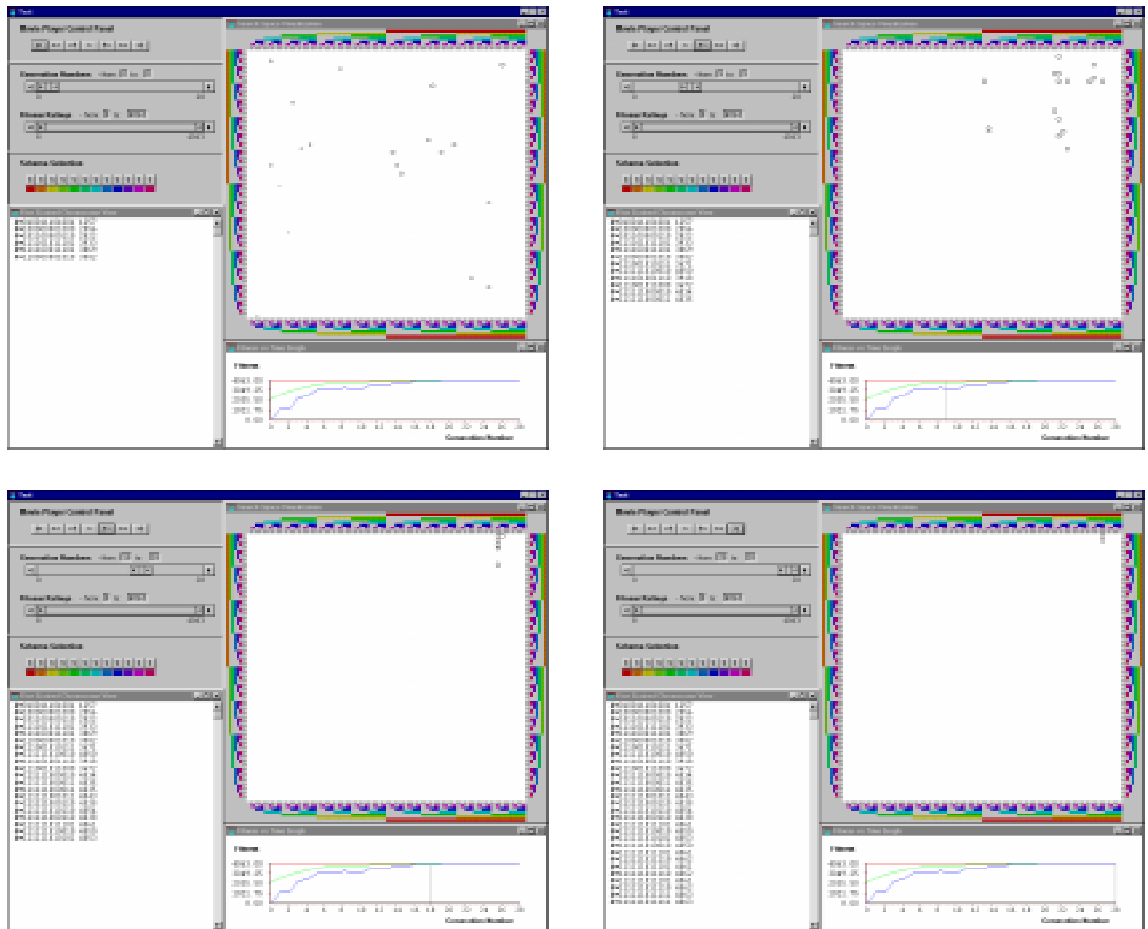


Figure 7.9: Four screen images taken from GONZO showing the population of a GA solving the 12 bit Maximum Integer problem after generations 0, 9, 18 and 28 (top left, top right, bottom left and bottom right, respectively).

the vertical axes, such that the worst organism (000000000000) is located at the bottom left hand corner of the search space view, the best organism (111111111111) is shown at the top right hand corner, the middle range organisms (101010101010) and (010101010101) are shown in the bottom right hand corner and top left hand corner, respectively. Figure 7.10 shows the complete GA run containing the chromosomes in every population.

7.4.2 The De Jong F1 Test Problem

A wide range of GA test problems have been proposed for studying the theory of GA search and GA design since Kenneth De Jong's original suite of test problems [De Jong, 1975]. However the De Jong suite of test problems is considered a classic set of problems and continue to be used by

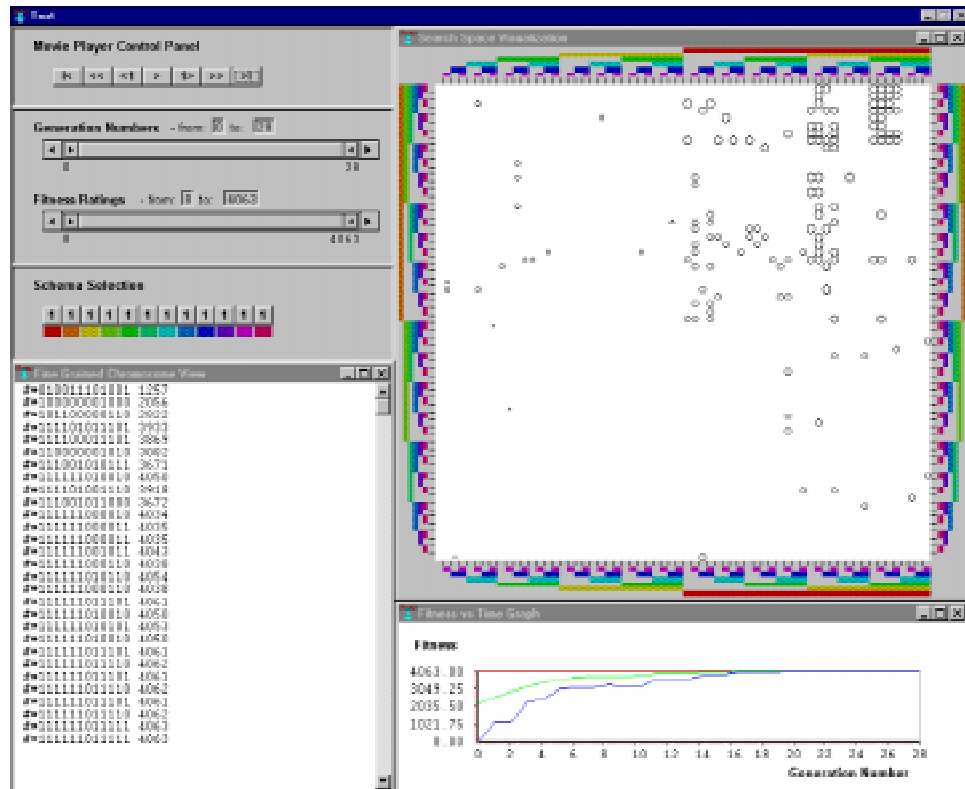


Figure 7.10: A screen image taken from GONZO showing the complete population data of a GA solving the 12 bit Maximum Integer problem. The size of each point shown on in the scatterplot view (top left) indicates the magnitude of the fitness rating for the chromosome at that position in the search space matrix. In this case the magnitude of the fitness ratings ranges from 0 at the bottom left hand corner of the search space matrix to 4095 ($2^{12} - 1$) at the top right hand corner of the search space matrix.

researchers exploring GAs. Having a complete fitness landscape² and investigating the GA's search path over that landscape under a number of different design conditions enables the user to investigate the evolutionary search behaviour of their algorithms and extract design guidelines based on the GA's behaviour under the test conditions.

The F1 test problem attempts to minimize the sum of the squared decimal values of three ten bit binary strings. Figure 7.11 shows the relationship between fitness (on the vertical axis) and two problem dimensions (variable2 and variable 1). Although this gives a strong indication of the relationship between the decimal values of the GA's chromosomes and their fitness ratings, the actual binary chromosome values are not shown. The three variables have values in the range -5.12 to +5.12

²A fitness landscape is a surface plot illustrating the variation in fitness across the entire search space.

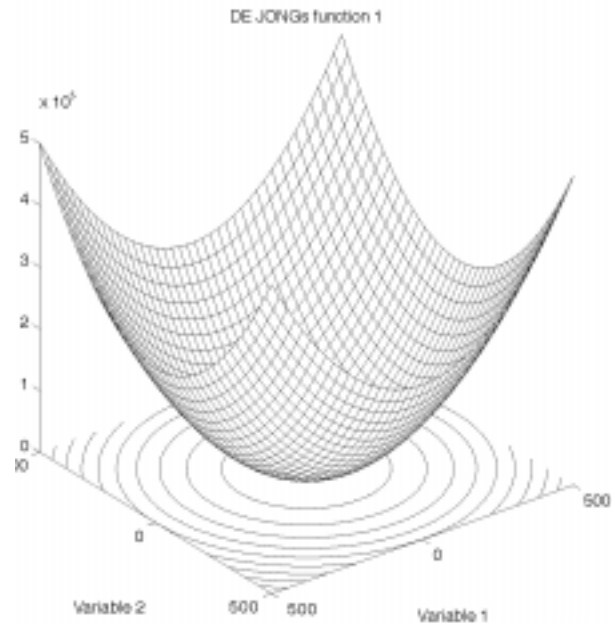


Figure 7.11: A 3D illustration of part of the fitness landscape of De Jong's F1 test problem. The decimal values of two of the three problem dimensions are plotted here on the x and z axes and the corresponding fitness rating is shown on the y axes of the 3D surface plot. This problem attempts to minimize the fitness rating such that the GA evolves solutions located around the centre of the above fitness surface.

represented as three ten bit binary genes. In this example the best chromosome value for each of the three problem dimensions is 0, however, the second best chromosome values are -0.01 and + 0.01, in terms of the GA's genotypic search space these values equate as follows; $0 = 1000000000$, $-0.01 = 0111111111$ and $+0.01 = 1000000001$. The large difference in genotypic space between 0 and -0.01 is not visually apparent in the 3D fitness landscape (phenotypic) view shown in Figure 7.11. However, the such differences are shown clearly in (genotypic) search space views (see Figure 7.13)

The search space visualization available in GONZO can be applied to show the complete fitness landscape of GA test functions, such as De Jong's F1 function, and illustrate the GA's search behaviour in the search space. Figure 7.12 shows a series of screen view examples illustrating the GA's evolutionary search path. The Lisp code used to create this view in GONZO is available in Appendix D.

Figure 7.13 shows how GONZO can be used to focus in on distinct regions of the fitness landscape. The top two images show how the complete set of all chromosomes considered during the GA's run can be filtered to show only those chromosomes within a fitness range of interest, in this case either

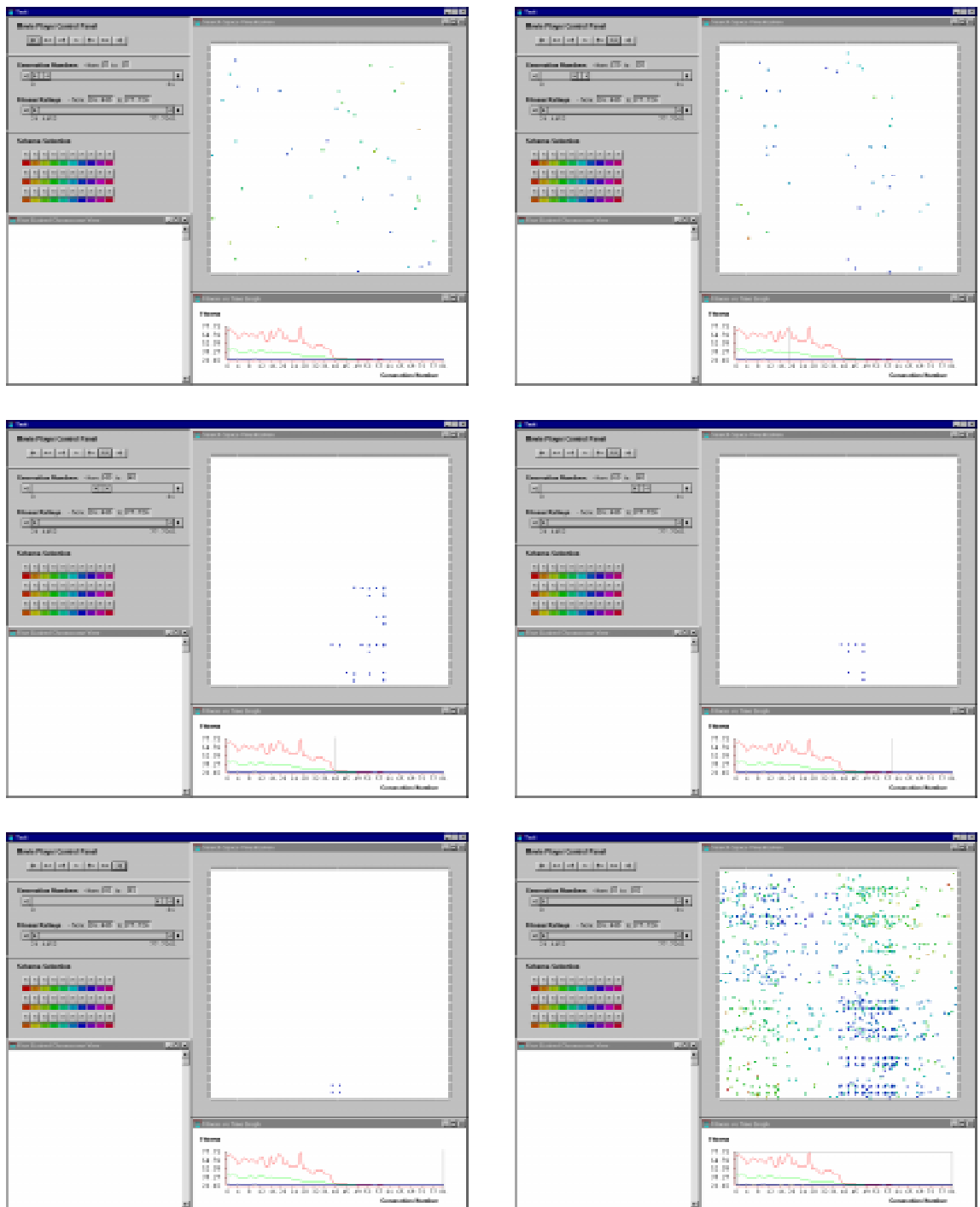


Figure 7.12: A series of example GONZO screen images for a GA solving De Jong's F1 test problem. The same screen layout is used here as in Figure 7.9, the state of the GA at generation 0 (top left), 20 (top right), 40 (middle left), 60 (middle right), and 81 (bottom left) are shown along with the complete GA's run (generations 0 to 81, bottom right). In this example each chromosome is illustrated by a rectangle, the colour of each rectangle indicates the corresponding chromosome's fitness rating. The colours range from red for a high (i.e. poor) fitness rating, to blue for a low (i.e. good) fitness rating. The projection order for the search space mapping is taken from the most significant bit for each problem dimension i.e. (0 11 21 1 12 22 2 ... 28 9 19 29).

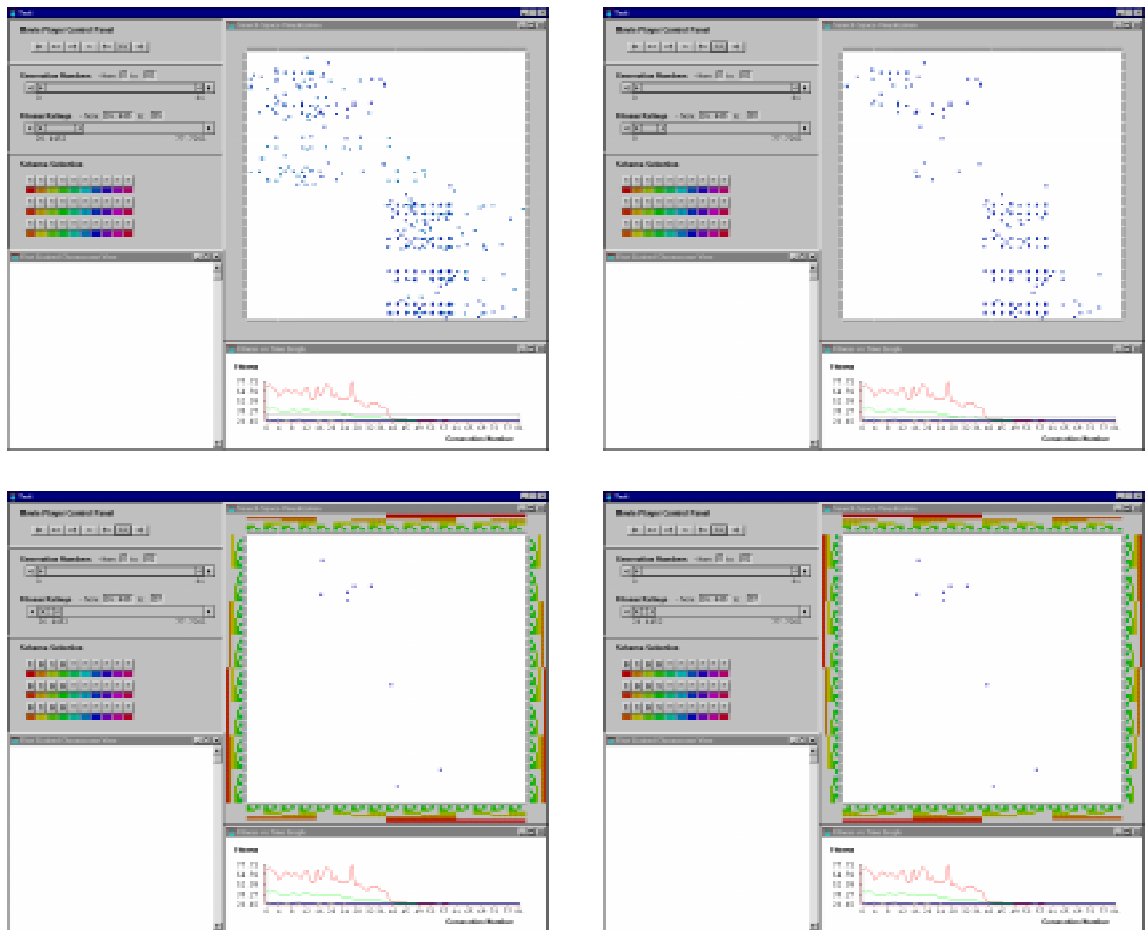


Figure 7.13: A set of fitness bound GONZO screen images for a GA solving De Jong's F1 test problem. Three fitness ranges are shown here, showing chromosomes within the ranges 26 to 35 (top left), 26 to 30 (top right) and 26 to 27 (bottom left and right). The bottom two screen images show two different schema selected by the user in the schema highlighting dialog (third left). These illustrate the different schema structures held by two near optimum solutions (shown in these two screen images as the lowest right-most coloured box and the highest left-most coloured box in the search space matrix). The same screen layout is used here as in Figure 7.12.

less than thirty five (top left), or less than 30 (top right). The second pair of images show how the schema highlight selector can be applied to highlight two different regions of the search space view. As a result, the user can examine the local structure of the GA's chromosomes.

7.4.3 The Royal Road Problem

The last example visualization shown here illustrate how Gonzo can be used to examine a GA solving a royal road function [Mitchell et al., 1991]. Royal road functions reward binary chromosomes for the number of building blocks they contain. The example used here is for a 64 bit binary chromosome which is split into eight 8 bit sections. For each of the eight 8 bit sections containing eight 1's, 8 points are added to the chromosome's fitness rating. For each of the four 16 bit sections containing all 1's, sixteen points are added to the chromosome's fitness rating. For each of the two 32 bit sections containing all 1's, 32 points are added to the chromosome's fitness rating. Finally for a 64 bit chromosome containing all 1s, 64 points are added to the chromosome's fitness rating. Therefore, the worst possible chromosome with a fitness rating of 0 is a 64 bit string containing no eight bit "all 1" sections, and the best possible chromosome with a fitness rating of 256 (i.e. $eight \times 8 + four \times 16 + two \times 32 + one \times 64$) is a 64 bit string containing only 1's.

Visualizing a 64 bit binary search space is a difficult task in that it contains 1.845×10^{19} solutions, a search space matrix of $4,294,967,296 \times 4,294,967,296$ is needed to represent each solution as a unique point in space, however the screen resolution currently available for displaying the search space (a maximum of 1280×1024 pixels on a common IBM PC) means that such a view becomes very reduced. One solution to this problem is to use a zoom and pan mechanism to explore the search space at a more reasonable level of granularity, but seeing the entire search space directly is not possible. An alternative solution, shown in Figure 7.14, is to split the search space into eight views showing each chromosome as eight points in eight 8 bit views, rather than one point in a single 64 bit view. With problems, like royal road functions, that evaluate the chromosomes based on the values of individual sections it is useful to examine how the individual building blocks are spread through the population.

In order to produce multiple search space visualizations from a single GA, a new method called the `search-space-visualization-matrix` is introduced here that re-

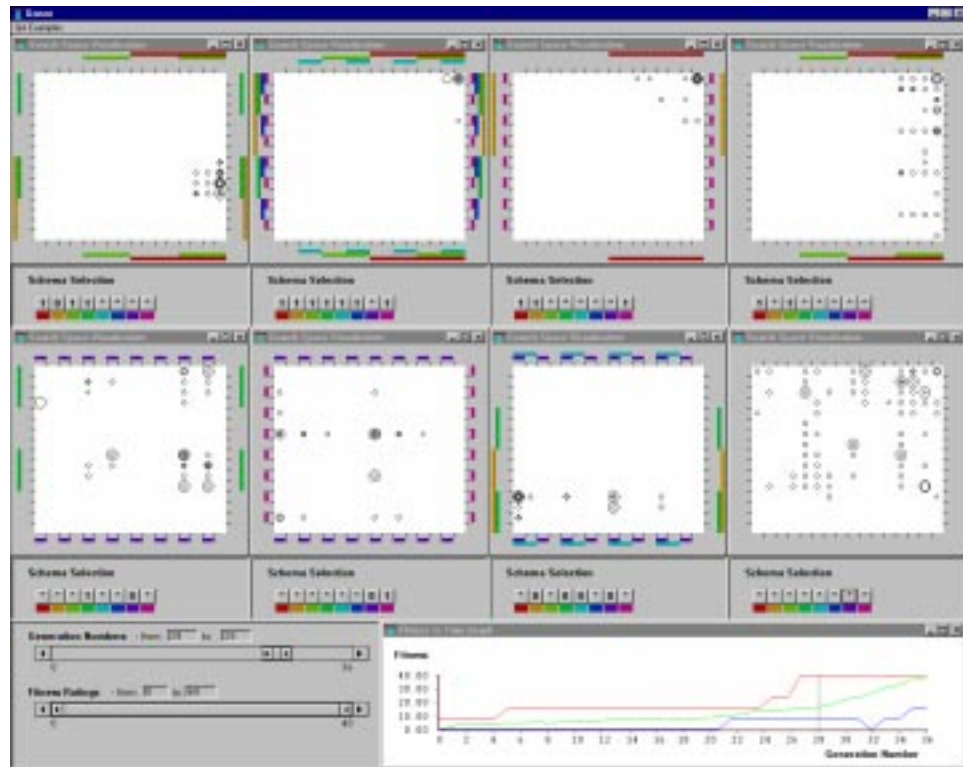


Figure 7.14: An example screen image taken from GONZO for viewing a single run of a GA solving the Royal Road function [Mitchell et al., 1991]. Eight search space visualizations are used here to illustrate the eight sections of the sixty four bit chromosomes. The four scatterplots at the top of the screen view represent the first four building blocks for loci 0 to 7, 7 to 15, 16 to 23 and 24 to 31, and the four scatterplots in the middle of the screen view represent the last four building blocks for loci 32 to 39, 40 to 47, 48 to 55 and 56 to 63.

uses the existing `search-space-visualization` method. The Lisp code for the `search-space-visualization-matrix` is shown in Appendix D, and is applied as follows:

```
(create-search-space-visualization-matrix
  list-of-names dataset chromosome-mapping-technique parent-dialog
  list-of-exterior-boxes coordinate-mapping-technique list-of-views
  list-of-projection-locus-orderings)
```

The `search-space-visualization-matrix` method is used to produce the visualization shown in Figure 7.14, as follows:

```

(create-search-space-visualization-matrix
  '(scatterplot-view-0 scatterplot-view-1 scatterplot-view-2
    scatterplot-view-3 scatterplot-view-4 scatterplot-view-5
    scatterplot-view-6 scatterplot-view-7) ;; list-of-names
  run-1 ;; dataset
  'GSM-D ;; chromosome-mapping-technique
  *visualization-dialog* ;; parent-dialog
  '((cg:make-box 400 204 619 402) (cg:make-box 619 204 839 402)
    (cg:make-box 839 204 1058 402) (cg:make-box 1058 204 1278 402)
    (cg:make-box 400 502 619 704) (cg:make-box 619 502 839 704)
    (cg:make-box 839 502 1058 704) (cg:make-box 1058 502 1278 704)) ;; list-of-exterior-boxes
  'D-GSM ;; coordinate-mapping-technique
  (list text-view-0) ;; list-of-views
  '((0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15) (16 17 18 19 20 21 22 23)
    (24 25 26 27 28 29 30 31) (32 33 34 35 36 37 38 39) (40 41 42 43 44 45 46 47)
    (48 49 50 51 52 53 54 55) (56 57 58 59 60 61 62 63))) ;; list-of-proj-locus-orderings

```

7.5 GUI Front End

Although GONZO strives to maintain a sufficient level of expressive power for GA users by providing high-level Lisp commands for producing individual visualizations, this approach suffers from a lack of usability for users unfamiliar with Lisp programming. As a response to this draw back an additional menu-based graphical user interface is introduced here as an optional front end for GONZO. This interface provides a system menu for selecting individual examples of GA applications (and their default visualizations), and a pop-up menu for setting view specific options, in this case for selecting the image mapping used in the search space visualization. The use of these two types of menu are explained in this section.

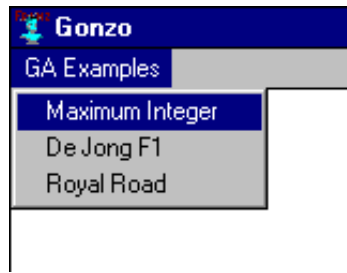


Figure 7.15: The system menu bar used in GONZO to select example GAs. Three options are available in this menu, these run the GA and present the visualizations for the maximum integer problem, De Jong’s F1 test problem, and the royal road problem (as described in Section 7.4).

7.5.1 GA Examples System Menu

Figure 7.15 shows the system menu available in GONZO. Three examples can be selected from the “GA Examples” menu: “Maximum Integer,” “De Jong F1,” and “Royal Road.” These options correspond to the three example GA applications described in Section 7.4, in each case the corresponding GA is run and the visualizations described in Section 7.1 are displayed. This menu relieves the user of the task of setting up the GA and writing the calls to the visualization functions described in Section 7.2. Furthermore, the Lisp code used to produce these examples gives the user an indication of how they may go about producing their own GA applications and alternate visualizations.

7.5.2 View Specific Pop-Up Menu

Figure 7.16 shows the pop-up menu that is included as part of the search space visualization. This pop-up menu appears when the cursor is within the display area of the search space matrix and the user presses the right hand mouse button. The menu contains two options; one to set the image mapping used by the search space view and a second to set the minimum size of the chromosomes’ images.

The “Image Mappings” menu option contains four further options which relate to the visual variables used to represent the chromosomes’ fitness ratings; namely, size, colour and value (as described in Section 5.2.2). The “Circle Size” and “Box Size” options set the image mapping used in the search space view to map the chromosomes’ fitness ratings to the *size* (i.e. area) of the circle or box used to identify each chromosome. The “Box Colour” sets the *colour* of the box to the chromosomes’ fitness

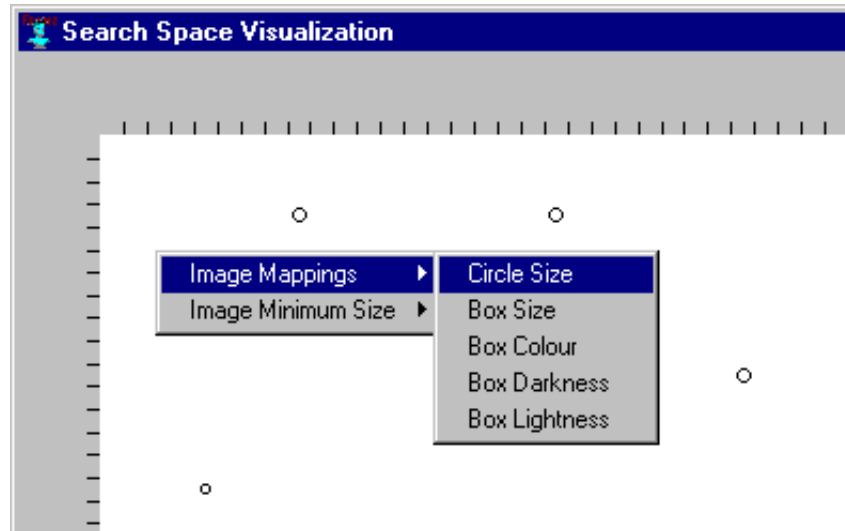


Figure 7.16: The pop-up menu bar used in GONZO to identify the image mapping used in the search space visualization. Five options are available in this menu for circle size, box size, box colour, box darkness and box lightness image mappings.

rating in the range of blue (for low fitness values) to red (for high fitness values). The “Box Darkness” and “Box Lightness” menu options map the chromosomes’ fitness ratings to the colour *value* of the chromosomes’ box images. These can be used to link the magnitude of each chromosome’s fitness rating to the darkness or lightness value of the corresponding box image. These two options enable the user to emphasise the chromosomes with large or small fitness ratings. These are important options needed when visualizing GAs that maximize or minimize the chromosomes’ fitness ratings: Box Darkness emphasises the chromosomes with high fitness values as they appear as dark boxes, and Box Lightness emphasises the chromosomes with low fitness values. In practice, these two options can also be useful for emphasising the fit and unfit regions of the search space considered by the GA.

The “Image Minimum Size” menu option allows the user to set the absolute minimum width and height in pixels that a chromosome icon be. Five options are available: 1, 2, 3, 4, and 6. This menu item is only available when the Circle Size or Box Size option is selected as the image mapping. The size of the chromosome icons for the box colour, lightness and darkness options is determined by the resolution of the search space view.



Figure 7.17: Getting started with GONZO. This figure shows the three stages involved in executing GONZO: (1) loading the `gonzo.lsp` file that creates GONZO, (2) starting GONZO with the command “(gonzo)” in the Lisp listener’s command line, and (3) selecting an example GA application from the GONZO system menu.

7.6 User Walkthrough

This section explains the individual steps involved in loading and running GONZO. This includes a walkthrough description of the steps to be taken in order to view the example GA applications presented in Section 7.4 with the visualizations described in Section 7.1, and the steps required to introduce other GA applications and alternate GA visualizations. The Lisp code used to produce the example GA visualizations included in Appendix D can be used as templates for introducing alternate GAs or alternate visualizations. Further information regarding the use of the GECO GA prototyping environment can be found in [Williams, 1993].

7.6.1 GA Examples and Their Default Visualizations

As noted in Section 7.2, GONZO is written in Lisp using the Allegro Common Lisp environment. In order to use GONZO, first start the Allegro Common Lisp environment and load GONZO (i.e the file called “`gonzo.lsp`”), this will load the necessary files which define GONZO. The menu based environment is started by typing the command “(gonzo)” at the Lisp listener prompt. This will open a new window that includes a system menu which will allow the user to select an example GA (see Figure 7.17). As noted in the previous section, the user can select either the maximum integer problem, De Jong’s F1 test problem, or the royal road problem. The corresponding GA will then be executed and the visualizations illustrated in Section 7.4 will be displayed.

In GONZO the default image mapping displays the chromosomes in the search space visualization as a circle, the size of each circle indicates the value of the corresponding chromosome’s fitness rating. The image mapping and minimum image size used in the search space visualization can be set using

the pop-up menu described in Subsection 7.5.2. Regions of the search space view containing individual schemata of interest can be highlighted using the schema highlighting dialog, and the GA's execution can be navigated using the movie player control panel and the generation and fitness range selector (as described in Section 7.1).

When the user is finished using GONZO they can close it in the same manner as they would close any other window: either by selecting the close button at the top right hand corner of GONZO's main window, selecting the window menu at the top left hand corner of the main window and choosing the "Close" option, or by pressing the "Alt+F4" key combination (also identified to the right of the Close option on the main window's menu bar).

7.6.2 Alternate GAs

Other GAs, not included in the GA Example menu, can be visualized using GONZO. In the case of the example applications the GA is executed and the result is stored in an instance of the `GECO ecosystem` class. The `GECO ecosystem` class includes slots for the run's population, number of generations, number of evaluations and genetic plan (as described in Section 6.1.1). GONZO uses the population-statistics slot of the ecosystem's population as its history module. Providing the user's GA is written in `GECO` and uses the population-statistics slot to record each generation's organisms, min-score, avg-score and max-score then the views, mappings and navigators defined above can be directly applied. In order to use the visualizations described in Section 7.1, the user can simply substitute the name of their own GA's ecosystem for the *dataset* variable used in each visualization's initialization command, as given in Section 7.2.

To introduce new GA examples to the GA Examples menu bar the user must first define a function that executes their GA and calls the appropriate visualization methods using the name of their GA's ecosystem class instance. A new menu item must then be added to the GONZO system menu, and the name of the user's new function should be given as the function to be called when the menu item is selected. Within GONZO the production of menu items is automated such that any menu labels and their corresponding function names, which are held in the `menu-components-list` and `menu-function-list` global variables (defined in the "gonzo-menus.lsp" file), are automatically created when GONZO is invoked.

7.6.3 Alternate Visualizations

Alternate visualizations can be introduced in a similar manner to alternate GAs. The user can create new visualizations in Lisp using the HENSON framework. The resulting functions can be executed either from the Lisp environment's command line, or through GONZO's system menu by adding new labels and function names to the `menu-components-list` and `menu-function-list` global variables.

7.7 Summary

This chapter introduced the design features of GONZO, illustrated how the design features could be specified using the HENSON framework, presented the specific Lisp implementation and application of the high-level GONZO visualization functions, and explained how GONZO could be applied to illustrate the search behaviour of a set of example problems.

GONZO is applicable to the majority of GAs, the only known exceptions are for GAs with more than one chromosome per genotype, with chromosomes containing continuous alleles, or for chromosomes with no fixed maximum length. Representations using more than one chromosome per genotype can be used in GECO but the search space view and mapping module of GONZO would need to be adapted to cope with the multiple chromosomes. Although continuous alleles are not a very common GA representation, GECO can be used to build algorithms with continuous alleles. However, continuous values cannot be represented as unique points using the extensive repartition technique deployed in the search space view. Replacing the *chromosome-mapping-technique* with a mapping method more suited to continuous data would alleviate this problem. The search space for genotypes with no maximum length are effectively infinite and therefore are difficult to map into a 2 or 3 dimensional scatterplot. However, providing the user can zoom in and out of the search space view this can be accommodated within the extensive repartitions translation technique.

To conclude this chapter the visualization attributes of GONZO are compared against the set of user requirements established in the user study (Chapter 3). All four issues, *usability*, *expressiveness*, *interactivity* and *supportiveness*, have been addressed (to a greater or lesser extent) in the development of GONZO.

Usability

Three different types of ready to use generic GA visualizations are available in GONZO; the coarse-grained fitness versus time graph, the medium-grained search space visualization, and the fine-grained chromosome view. These three linked representations enable the user to obtain an overview of the GA's evolution using the fitness versus time graph, zoom and filter information of interest using the search space visualization and its associated navigators, and select and view details of individual chromosomes using the fine-grained chromosome view. Since this system was developed this approach has been summarized as the “visual-information-seeking mantra” i.e. “Overview first, zoom and filter, then details on demand” [Shneiderman, 1998, page 523].

GONZO includes a set of three example GA applications written using the GECO GA prototyping tool [Williams, 1993]. These are available from a drop-down “GA Examples” system menu which runs the corresponding GA and presents an interactive off-line visualization of the GA's execution. The system menu enables people to use GONZO without writing any Lisp commands. However, in order to apply a GA to a new problem the user will have to define an appropriate problem representation, write an evaluation function to evaluate their GA's chromosomes, and define a suitable set of selection and reproduction operators (as explained in Section 2.2.3). These programming tasks are facilitated in this case by the use of the GECO GA prototyping framework. The user will also have to program in order to introduce any new visualizations, in this case the HENSON framework facilitates the task.

Expressiveness

Additional visualizations can be introduced by the user at practically any level of programming abstraction, from graphics programming in Lisp, through history, view, mapping, and navigator descriptions in HENSON, to view configuration and re-use in GONZO.

Interactivity

Navigation dialogs are available in GONZO to explore the GA's search sample for individual generations as well as ranges of generations and ranges of fitness ratings. Further investigative interaction enables the user to analyse the search space independently of the GA.

Editing the GA's parameters and components is possible via the command line in GECO. No

additional views or navigators were included within the design of GONZO to support further interactive online algorithm editing, but this would be a relatively trivial extension.

Although the use of direct manipulation to edit the chromosomes in the population is potentially possible within the search space visualization this feature was not included in the design of GONZO. The focus of GONZO was to support the user's understanding of the GA's search behaviour rather than guiding or intervening in the evolutionary process. Introducing a direct manipulation navigator into the search space visualization would be one way of enabling the drag-and-drop direct manipulation of chromosomes. This would simply re-use the *coordinate-mapping-technique* to remove the chromosome identified by the cursor coordinate when the "mouse-down" event is recorded and replace it with the chromosome identified by the cursor coordinate at the next "mouse-up" event.

Supportiveness

Finally, GONZO provides an extensive degree of support for the user's understanding of the GA's exploration of the search space. The user's sense of position within the GA's run is supported by the fitness versus time graph; the user's sense of the GA's sampling of the search space is supported by the search space visualization, and any further details regarding individual chromosomes in the population or unexplored regions of the search space can be viewed in the fine grained chromosome view. In fact, supporting the user's understanding of the GA's search behaviour is the explicit intention of GONZO (see Section 7.1).

The second form of support identified in the user study, i.e. design support, is not provided directly by GONZO. This is an important and unfulfilled need of the GA community that requires additional support to that of SV. The provision of design support for GA users, and the role that visualization can play in design, is discussed further in Section 8.3.

Chapter 8

Discussion

This thesis began by proposing SV as a method for alleviating the “black box” image associated with the application of EAs. It was suggested that EA users do not understand the search behaviour of their algorithms and therefore find it difficult to make design modifications or guarantee the quality of the solutions found. This thesis has introduced a number of search space representations, based on Sammon mapping and extensive repartitions, which enable the user to see their GA’s sampling of the search space. This in effect removes the lid of the GA’s black box.

The validity of the initial motivating problem (i.e. the black box approach) was supported and further characterized by the findings of the user study. The level of support currently available from existing GA and SV systems was explored and a number of favourable features were derived. The design rationale behind this project, including the principled design of visualizations, the advantages of an open and extendable framework approach and the development of high-dimensional search space representations were discussed. As a result an extendable framework for supporting the development of GA visualizations was produced and an example visualization tool for illustrating the search behaviour of GAs was developed.

This chapter discusses the limitations of this work, explores how the results may be applied to other forms of EA, and speculates on some of the consequences of involving SV within the GA development and application lifecycle.

8.1 A Critique

This section discusses the findings made during the course of this project - specifically, the validity of the original motivating problem, the implementation of a principled design approach, the usability of the HENSON framework, and the difficulties incurred while visualizing high dimensional data.

8.1.1 The Validity of the Perceived Problem

Exploring the working practices of GA users in order to examine the problems that they encounter could be seen as an ineffective study, given that the problems they encountered were never so severe as to dissuade them from using GAs. Perhaps it would be more valid to survey people that at sometime had attempted to use GAs but had found them unusable.

However it could also be argued that the amount of effort that anyone is willing to spend in understanding something is directly linked to their perceived pay-off of the results. The difficulties experienced by GA users may be equated to the difficulties of GA non-users, the difference between the two groups being the perceived benefits of overcoming these difficulties. Furthermore, the limited level of insight afforded by those non-users would be of little help in designing visualization support for the real users of GAs.

The results of the study did in fact validate the perceived problem. Very few of the respondents explored their algorithm's search behaviour, or took any additional steps to verify the results that their algorithms found, see Section 3.2.6 finding number 8. The respondents typically used default algorithm designs and parameter settings, and any changes they made were done through trial and error, see Section 3.2.6 findings 6 and 7. However, this lack of investigation was not indicative of a lack of interest, as the respondents were very interested in seeing the chromosomes in each population. However they had no effective means of viewing this information (finding 9). Summary measures regarding the diversity of the population, such as similarity measures, were also considered useful but again difficult to produce (finding 10).

The GA user study established the validity of the perceived problem and provided an additional insight into the working behaviour of GA users and their opinions regarding the use of interactive GA visualizations.

8.1.2 Implementing a Principled Design Approach

The graphic design principles put forward by Bertin and Tufte are regularly cited as a source of guidance for visualization design, but many visualizations are still produced using “unfriendly graphics” [Tufte, 1983]. The most common errors regularly made are the inappropriate use of the six retinal variables (size, value, texture, colour, orientation and shape), and poor legibility either through dense graphics, poor angular separation, or poor retinal separation (see Section 5.2.3). Practicing good graphic design is more difficult than simply citing it. However, re-usable visualization libraries provide a solid start, and the supplied visualizations are exemplars, from which the user can gain an understanding of the benefits of good graphic design. Furthermore, through the inheritance and specialization of these visualizations, the user can produce new visualizations that contain the graphic design features of their parents.

8.1.3 The Usability of the Framework Approach

Any framework essentially provides the user with a series of structures for building new things, in this case providing a series of visualization structures. The HENSON framework enables the user to specify their visualization in terms of players, views, mappings and navigators. As a result the user can avoid a lot of the low-level graphics programming associated with visualization.

The HENSON framework provides the user with an object-oriented view hierarchy from which they can select and apply a view to produce a standard visualization, or inherit the attributes of an existing view and adapt them to produce a new view. In addition to re-using views, the navigators, mappings and generic players identified in HENSON can be re-used to produce a range of visualizations. Although HENSON reduces the effort required to produce new visualizations, it requires the user to be able to program in an object oriented fashion using Lisp.

8.1.4 Really High Dimensional Visualizations

One of the possible criticisms of search space visualization is the sheer size of the search space. Showing the chromosomes in a search space as unique points in a display area is not possible when the resolution available in the display area is less than the resolution of the search space. Given the size of most GA search spaces, it could be argued that viewing such spaces would provide only a

gross indication of the GA's search behaviour.

This is a valid criticism and the very reason why it is so important for the user to be able to navigate the search space. By using a zoom mechanism for example to increase the resolution of the search space view, the user can begin to appreciate the scale of the search space. The size of a GA's search space generally is incomprehensible, explicitly displaying the scale of a search space view and the view's relative coverage and location within the search space, enables the user to appreciate the actual size of the search space and comprehend the more relevant parts of it.

8.2 The Contributions of This Case Study

This section reviews the contributions of this thesis and explores how the work carried out here may be applied to the rest of EC and SV. The foundation of this project, the GA user study presented in Chapter 3, is the only known empirical study of the working practices of GA users. Not only was the insight gained through this study important to this project it also provides a foundation for the design of future GA environments. Furthermore, it gives an initial insight upon which future empirical studies can build to investigate further the work of GA users, as well as a basis for comparing the working practices of users of other forms of EC, such as EP and ESs.

The review of the related work given in Chapter 4 is one of the first known attempts to provide a comprehensive overview of GA visualization. This is currently a very new area of EC, borrowing heavily from its parental fields of software and information visualization. As noted at the start of this project interest in the visualization of EC is growing, the usefulness of "state of the art" reviews, such as the one presented here, provide a valuable insight to those seeking an introduction to the area.

The design rationale adopted in this project is a problem independent approach for producing visualizations based on the principles of graphic design (Chapter 5). Bertin's Semiology of Graphics was used as a basis for designing GA visualizations. The application of this approach produced some new 2D search space representations using Sammon mapping and extensive repartitions. The investigation of Sammon mapping as a means for producing search space visualizations, was carried out in collaboration with Richard Dybowski and Peter Weller [Dybowski et al., 1996]. This was the

first method effectively to produce high dimensional search space visualizations. Since this work was published the application of Sammon mapping as a technique for producing GA visualizations has been further investigated by Hartmut Pohlheim [Pohlheim, 1998].

At the time the extensive repartitions technique was developed for visualizing a GA's search space, it was thought that this approach to producing low dimensional maps of high dimensional spaces was a new one, see [Collins, 1996] and [Collins, 1997]. However, as noted in [Collins, 1998] this was not the case, as previously explained this approach first appeared in 1782 and has recently been used as a technique for information visualization [Mihalisin et al., 1991]. However, even though this technique is an old one, its application to this domain is new, the use of a direct translation function for converting chromosomes to coordinates is not presented elsewhere. Furthermore, the use of an interactive schema highlighting dialog is a new approach for identifying values in this representation, a feature which enables the user to identify the contributions made by different chromosome schemata.

The principled design approach adopted here is an effective means for guiding design to ensure the development of effective visualization. However, the disembodied rules for exploiting the properties of the graphic system in order to illustrate information, are not easily applied. The HENSON framework presented in Chapter 6 provides a hierarchy of GA specific views which should embody the graphic design principles of Bertin and Tufte. Thereby enabling the user to re-use good visualization designs and experience the benefits of good graphics without having to pay any of the costs associated with designing or programming. Furthermore, the structure of the HENSON framework encourages the user to identify the information of importance for answering any queries they may have. This is achieved by requiring the identification of a visualization's players and those players' important events during the course of the GA's run. In this way HENSON not only facilitates the development of GA visualizations, it also encourages the users to think about the queries they are asking and analyze the information required to answer those queries.

Finally, the GONZO visualization tool is a flexible, generic GA visualization tool designed to illustrate the search behaviour of any GA that uses a categorical coding alphabet. The navigator, mapping and view components of GONZO can be applied in any way the user chooses and can be further extended through the use of the HENSON framework. In terms of the scope of the GONZO tool, any categorical dataset can be represented. Therefore, providing a category-based representation is

used, the search behaviour of any GA, EP or ES can be represented by the extensive repartitions technique and displayed using GONZO.

The generic contributions made here that can be deployed to other aspects of visualization and software development can be summarized as follows:

- An insight into the working practices of GA users.
- An approach for producing visualization support based on the problem investigation, the identification of relevant information, and the principled design and development of appropriate visualizations.
- The HENSON GA development framework for developing visualization and interaction support for GA users.
- The GONZO visualization tool - a tool for exploring discrete high dimensional GA search spaces.

8.3 Future Work

This final section discusses some of the future projects that will follow on from the work described here. Two important conclusions, drawn from the GA user study, were that some GA users find it difficult to design GAs and are unable to follow their GA's search behaviour. This thesis has addressed the second of these two problems through the analysis of the users' working practices and the principled application of graphic design to produce effective search space visualizations. However, little effort has been expended here on the provision of GA design support. Subsection 8.3.1 attempts to reconcile this by exploring how existing design support techniques could be applied to the GA domain. Although the provision of design support is beyond the scope of this project, SV has a role to play in supporting the design task and this is also discussed here.

Returning to the future work regarding the use of SV to support peoples use of EC, Subsections 8.3.2 and 8.3.3 investigate the consequences of enabling the user to interact with their visualizations. Subsection 8.3.4 explores the future development of additional search space representation techniques particularly for real valued chromosomes. Finally, Subsection 8.3.5 discusses the continued development of the HENSON framework and its application to other domains.

8.3.1 Supporting GA design

The application of design support technology to the GA design process, although beyond the scope of this project, is recommended as a method for supporting the GA design process, and is proposed as an important future project. This subsection highlights some of the design problems GA users face, as identified in the user study (Chapter 3), and briefly discusses how some of the current knowledge engineering approaches could be used to support the design process along with the role SV could play within such a supportive design environment. Currently there are no known design support systems that specifically support the GA design process.

The Problem with GA Design

According to the findings of the GA user study (Chapter 3), two distinct problems currently face GA designers. First, the complexity of the problem domain and the designer's understanding of that domain makes it difficult to represent the problem sufficiently or evaluate the proposed solutions (i.e. chromosomes) effectively. Complex problems which are difficult to understand, even by those working in the domain, are inherently difficult to describe using an abstract representation, such as a string of binary symbols (see Tables B.8 and B.9, page 265). Even when a representation has been defined evaluation is still a difficult task, not because of the chromosomes' abstract form but simply because of the complexity of the problem domain (see Table B.12, page 268). Furthermore, when a sufficient understanding of the problem domain is achieved the issue of credit assignment becomes a problem. It is important to assign more credit to "excellent" solutions than "good" solutions but not to the extent that good solutions are lost.

Secondly, the designer's appreciation of the contribution made by their GA's components and parameter settings is hampered by the fact that they are unable to observe their GA's search behaviour, and thereby judge the contribution made by different component configurations or parameter settings. As a result, GA users typically reuse what worked for them in the past, or adapt any available guidelines from the literature (such as [Davis, 1991] or [Goldberg, 1989], see Tables B.7, B.8, and B.9). None of the respondents reported using any theory-based approaches for determining either the algorithm components or their parameter settings. Rather the design practices evident from the results of the GA user study indicated an empirical approach to algorithm design. This approach is

essentially reliant on expert knowledge gained through the personal experiences of the designer.

Design Support Systems

A variety of knowledge engineering approaches are available to support the design process, such as case-based reasoning, constraint satisfaction and heuristics problem-solving methods. For reasons of brevity, only case-based reasoning is explored here. Case-Based Reasoning (“CBR”) is a general paradigm for problem solving, based on the recall and reuse of specific experiences. This approach is frequently used for supporting the design process by reminding designers of previous experiences that can help with new situations [Maher and de Silva Garza, 1997]. There are two major considerations within CBR in design; the representation of design cases, and the process models for recalling (i.e. accessing) and adapting design cases.

Representing design cases involves the abstraction of the experience into a symbolic form (for further details see [Kolodner, 1993] Chapter 5). Developing a case-based system for GA designers is a non-trivial task. The representation of design cases and the process models for recalling and adapting design cases must be sufficiently general to include any potential GA design for any problem domain. The key issue here is to identify the information within a design that facilitates its re-use i.e. the key elements that provide a complete and sufficient description of the design and its application. Generic GA systems, such as GECO (Genetic Evolution through the Combination of Objects) [Williams, 1993], GENESIS (GENETic Search Implementation System) [Grefenstette, 1984], GENOCOP (GENetic algorithm for Numerical Optimization for CONstrained Problems) [Michalewicz, 1991] and GALIB (a C++ Library of GA components) [Wall, 1996], are used by both industry and academia for building GAs (see [Ribeiro et al., 1994] for a review of some typical GA programming environments). These systems provide high-level GA-specific commands that could be used in a case-base to form a design specification. The GA’s output could also be recorded and used as part of a description of the design’s outcome.

Design-case recall involves the indexing, retrieval and selection, of design cases. When developing indexing and retrieval schemes [Maher and de Silva Garza, 1997] highlight the importance of flexibility for allowing the design specifications to change or be refined, and the abstraction of cases to decompose a problem specification into sub-problems in order to find a relevant design case or sub-

case. Case retrieval is generally done either informally by the user browsing through the design cases and selecting an appropriate case, or formally by defining a specification format and using pattern matching to retrieve appropriate cases.

Informal approaches to retrieval support the need for flexibility in developing an understanding of the new design problem through browsing existing designs. However, the size of the case-base and the richness of the indexing scheme determine the effectiveness of this approach. Case-base browsers are typically implemented as hypertext browsers with some capability for word search. Formal approaches are commonly implemented using attribute-value pairs to specify a new design problem, the attribute-value pairs are then matched against those held in the case-base and a weighted sum of matching pairs is used to propose a set of related designs. Other methods reviewed by [Maher and de Silva Garza, 1997] include retrieval based on function, retrieval based on matching images or gestalts, retrieval based on a hierarchy of problem specifications, or retrieval using a graph based representation of behaviour.

Finally, the process of *design-case adaption* for producing new designs involves three stages; propose, evaluate and modify. [Maher and de Silva Garza, 1997] describe three categories of design adaption based on who or what performs the adaption; the human designer, a knowledge intensive computational method, or a knowledge-lean computational method. Human design-case adaption leaves design changes to the user rather than the machine, this approach in effect produces a case-library i.e. a repository of information about designs, rather than an automated means for producing new designs. Knowledge-intensive computational case adaption methods include; constraint satisfaction, heuristic modification rules, subcase replacement, or model-based reasoning (i.e. design fixes). Rather than letting the designer actively employ their own knowledge of the problem domain to adapt a related design, knowledge-intensive methods use generic domain knowledge such as constraints, rules, plans, or models to guide the machine's formulation of new designs. Thirdly, knowledge-lean computational case-adaption attempts to use less knowledge intensive design search algorithms, such as GAs, to adapt the retrieved designs. Knowledge-lean methods use domain knowledge, in the form of constraints and design-problem specifications, to evaluate the potential of each adapted design solution.

The case-based approach matches well with the current working practices of GA designers. Al-

though GA users reported reusing their algorithm components and parameter settings, none of the users reported physically recording their design cases or recalling their previous designs from a case-base. By adopting this approach the GA designer is no longer reliant on their ability to remember previous designs, rather their designs are recorded and recalled for them. Moreover, by providing a case-base of design solutions to problems as yet unencountered by novice designers, the novice's initially steep learning curve may be reduced. Finally, through the provision of a shared case-base, either shared via an intranet within a company, or shared via the internet throughout the EC community, an extendable "living" design case-base may be made widely available.

Artificial Intelligence Supported GA Design

In the above description of case-based systems two major considerations were identified; the representation of cases, and their subsequent recall and adaption. EC is a domain that exists within a computer, the design of each algorithm is, therefore, already represented in the computer code used to define it. Hence, as previously described recording an algorithm's design and its outcome is a relatively trivial task. The only additional information required to produce a case-base is to provide a description of the original problem, the designer's decisions, reasoning and justifications for the proposed solution, and an explanation of the outcome. For certain problems this may amount to a lot of information, however, the predicted benefits outweigh the cost. These benefits include improved usability (i.e. a reduced cognitive load for expert designers recalling previous designs and for novice designers discovering other people's designs), time savings (i.e. savings in the time rewriting rather than recalling similar algorithm designs), and improvements in design documentation (i.e. explicit design documentation from case notes).

Recalling the design cases from a case-base, indexed for example by problem domain or chromosome representation, could be done manually by the designer, or with the help of an automated keyword search through the problem specification for each case. Case adaption can either be done manually by the designer, or automatically using knowledge intensive or knowledge-lean adaption. Within the GA community manual adaption is typically used (see the GA user study, Section B.2), however knowledge intensive adaption could be applied for problems in which a sufficient body of design knowledge can be expressed. For example, using design heuristics for the traveling salesperson problem (see

[Michalewicz, 1996]). The use of knowledge-lean design adaptation is already used in EC, algorithm adaptation occurs by including the algorithm's parameters within the chromosome's representation. Several examples can be found in [Baker, 1985], [Baeck, 1992], and [Hoffmeister and Baeck, 1992].

Finally, one of the issues to note within EC design is that a number of alternative algorithms can be applied to solve the same problem, in such cases there is rarely any single optimal design solution and therefore, the weighting applied either to any explicit design knowledge, or case retrieval methods, should be able to recommend alternate designs.

The Role of SV in GA Design

As previously noted online visualizations are made "on the fly," while the computer program is executing, where as offline visualizations are made "post-mortem," after the algorithm's execution. The only difference between these two forms of visualization is that one is produced using the current state values of the program and the other is produced using a recording of the state values. When it comes to applying SV technology to support the GA design process, offline visualizations may be used to facilitate the user's understanding of previous algorithm designs and their outcome.

Although CBR is described in terms of design case representation and recall, these two considerations are not easily separated. The recall of previous cases is made through the representation used. When SV is introduced as part of the case representation it also becomes part of the case recall. The designer is able to browse both the design cases' textual descriptions and visualizations. Providing the visualizations used improve the user's understanding of the design, the introduction of offline SV into the design support system gives an immediate improvement to the usefulness of the system. The degree of improvement introduced by adding SV clearly depends on the efficacy of the visualizations used.

Within GA design understanding the implications of the various design actions is particularly difficult. Not only do the different algorithm components interact producing an emergent behaviour, i.e. evolution, they interact in a stochastic manner such that any two executions of the same algorithm may well produce different results. Another thing that compounds this problem is the level of design feedback. If the designer's only form of feedback is the best result achieved by the algorithm, i.e. the best chromosome, then there is little information that they can use to inform future design. However,

if visualizations are available showing the search behaviour of the previous cases, then these can be used to explain the behaviour of the algorithm and inform future design. The level and format of explanation required by the designer may well vary from one designer to the next, therefore, the choice of visualization used to explain the GA's execution should be left to the designer. Provided the case histories record the GA's execution at a sufficient level of granularity then practically any visualization should be possible.

In the case of the HENSON framework, the History module can be used to produce either online or offline visualizations. Accessing the case base of GA designs through the History module would require no additional effort providing the outcome of the previous cases were recorded as History data files. Furthermore, within HENSON the designer's choice of visualization can be altered at any time by editing any of the view or mapping components. The search space visualizations described in Section 5.3, and the GONZO visualization tool described in Chapter 7, are directly applicable for producing representation of previous (i.e. recorded) GA design cases.

8.3.2 Human-EA Interaction

Interacting with a visualization is important so that the user can control the visualization in such a way that it enables them to answer the questions they have regarding the behaviour of the software. Interaction can be carried out at the level of *configuration*; changing the views to suit the subject of interest, *navigation*; controlling the displayed position in time or space, or *intervention*; directly altering the state of the underlying system. All three forms of interaction are useful in EC visualization, for example, configuring different view combinations, navigating through the generations of an EA's run or the structure of its search space, and changing the algorithm's components. The visualization work presented in this thesis opens the door for a whole new set of possible interaction opportunities.

An interactive search space visualization tool could be used to create or edit the EA's population. Using such a tool the user could not only set the initial population, but also re-introduce diversity or guide convergence toward interesting areas in the search space at any stage during evolution. However, further study is required in order to establish when user intervention is beneficial to the EA and when it is damaging.

8.3.3 Interactive Evolutionary Algorithms

Interactive Evolutionary Algorithms (“IEAs”) are a sub-set of EAs whose origin has been attributed to Richard Dawkins book “The Blind Watchmaker” [Dawkins, 1986]. The use of IEAs can be described as a two stage process in which the user is first shown each individual in the population in an appropriate (problem specific) form and then asked to evaluate each individual based on its perceived merit [Venturini et al., 1997]. Essentially the user takes on the role of the evaluation function, thereby removing the need to formally specify the problem evaluation criteria. This approach has been applied to several novel problems such as graphic art [Todd and Latham, 1992], music [Nelson, 1993] and knowledge discovery in databases [Venturini et al., 1997].

The current use of IEAs limits the user’s view of the search space to those points sampled by the current population and provides no information relating the chromosome’s genotypic structure to the features of the resulting phenotypic representation. Therefore, the user is unable to judge the chromosomes’ schemata and how those schemata may influence future solutions. By combining the first step in this process (i.e. displaying each chromosome in an appropriate form) with a visualization of the search space, the user can see each chromosome’s place in the search space and judge an individual not just on its phenotypic appearance but also on its genotypic structure, and ability to contribute toward new solutions. Thus enabling the user to see the “bigger picture” outside of the sub-space sampled by the current population. Furthermore, by identifying the contribution of individual genes within the chromosomes the user could choose to fix specific gene values and continue to evolve others, this may prove to be a useful way of testing and applying domain knowledge. Further investigation is needed into the use of search space visualizations for IEAs and the management of domain knowledge and user insight.

8.3.4 More Search Space Representations

One of the key findings of the GA user study was the need to visualize the GA’s search space. However, this is fundamentally a very difficult task, representing large high dimensional spaces on a two dimensional display screen requires the careful scaling of information. One technique considered here was Sammon mapping, this iterative technique scales the information held in the high dimensional space by attempting to preserve the relative Euclidean distances between all the points in both the

high and low dimensional spaces. Producing this mapping is time consuming and was ultimately dropped in favour of the extensive repartitions technique used to produce the search space matrix used in GONZO. However, the search space matrix technique is only applicable for discrete coding alphabets, EA's can also use real valued coding alphabets and therefore, further work is needed to produce search space representations for real valued chromosomes.

8.3.5 The Continued Development of HENSON

The HENSON framework developed during the course of this project, is an extended version of the VIZ framework. However, not all of the HENSON framework was implemented during the course of this project, future work will concentrate on the continued development of HENSON's view and navigator modules and its application to other domains.

Bayesian belief networks are applied in the areas of data analysis, knowledge discovery and machine learning and are based on the principles of bayesian statistics. Bayesian belief networks represent the relationships between a set of data variables as the links in a network. Given a specific network and a specific dataset, a bayesian belief network can be applied to model the dependencies between the linked variables. Then, for any given subset of variable values, the model can be used to predict the values of the missing variables. Bayesian belief networks typically model datasets of discrete variables and therefore, the extensive repartitions technique applied in this project may also be applied to view the variation of bayesian belief networks.

The work done here on the HENSON visualization framework and the search space visualization is to be applied to the visualization of bayesian belief networks, this work is to be carried out within The Open University's Bayesian Knowledge Discovery project (BKD), see <http://kmi.open.ac.uk/projects/bkd/>

Glossary

Allele - one of the (usually two) alternate forms of a gene	41
Alphaslider - a dynamic query interface widget used to identify an item or a range of items within a range. A range-defining alphaslider is similar to a scroll bar. However, rather than dragging a button to change a position within a range, a central bar can be directly manipulated to change a range within a dataset (see Section 4.39)	125
Associative Perception - if the eye can immediately reconstruct the uniformity of a series of undifferentiated points forming a uniform area, in spite of a given visual variation, this variation is associative. Shape, orientation, colour, texture and the planar dimensions are associative visual variables	136
Chromosome - the rod-like or thread-like structures found in the nuclei of the cells of living organisms that carry genetic information in the form of genes [Hawkins and Allen, 1991]	41
De Jong's F1 Test Problem - a GA test problem that attempts to minimize the sum of the squared decimal values of three ten bit binary strings in the range -5.12 to +5.12	198
Deme - a local population of closely related items	41
Dissociative Perception - if the eye cannot immediately reconstruct the uniformity of a series of undifferentiated points to form a uniform area, due to a visual variation, this variation is dissociative. Value and size are dissociative visual variables	136
Dynamic Query Interfaces - apply the principles of direct manipulation to database query methods in order to support information visualization. Dynamic query interfaces are characterised by their visual presentation of a query's components and results, rapid incremental and reversible control over a query, selection by pointing rather than typing, and immediate and continuous feedback	124

EAs - see Evolutionary Algorithms	33
EC - see Evolutionary Computation	33
Elevation - the use of the six retinal variables in an image - size, value, texture, colour, orientation and shape	135
EP - see Evolutionary Programming	42
ESs - see Evolution Strategies	44
Euclidean Distances - the distance separating two points in space. Calculated as the square root of the summed, squared differences between the coordinates of a point, in each of its dimensions (see Section 5.3.3)	146
Evolution Strategies (“ESs”) - a type of evolutionary algorithm developed by Ingo Rechenberg and Hans-Paul Schwefel based on models of evolution describing the evolution of individuals (see Section 2.2.2)	44
Evolutionary Algorithms (“EAs”) - a subfield of evolutionary computation which is made up of a number of powerful generic search algorithms, based on the guiding evolutionary principle of “survival of the fittest”	33
Evolutionary Computation (“EC”) - the study of a range of computing techniques which are based on models of biological evolution	33
Evolutionary Programming (“EP”) - a type of evolutionary algorithm developed by Lawrence Fogel, Alvin Owens and Michael Walsh based on models of evolution describing the evolution of species (see Section 2.2)	42
Extensive Repartitions - the repeated partitioning of an axis in order to illustrate classifications . . .	147
Finite State Machine (“FSM”) - a machine defined in terms of a finite alphabet of possible input symbols, a finite alphabet of possible output symbols, and some finite number of possible different internal states [Fogel et al., 1966, page 12]. FSMs are used in evolutionary programs as a means for describing an individual predictor. Evolutionary programs evolve populations of FSMs	42
Fitness Landscape - a plot illustrating the variation in fitness across an EA’s search space, typically displayed as a 3D surface plot showing the variation in fitness over two of the problem dimensions, e.g. Figure 7.11	100

- Fitness Versus Generation Number Graph** - see Fitness Versus Time Graph 86
- Fitness Versus Time Graph** - a 2D line graph commonly used in evolutionary computation to illustrate the variation in a population's fitness ratings (plotted on the y axis) over time as indicated by the evolutionary algorithm's generation number (plotted on the x axis) 86
- Geco** - the Genetic Evolution through Combination of Objects GA prototyping framework [Williams, 1993] 161
- Gene** - a unit forming part of a chromosome that determines a particular characteristic of an individual 41
- Genetic Algorithms ("GAs")** - a type of evolutionary algorithm developed by John Holland based on models of genetic evolution (see Section 2.2.3) 46
- Genotype** - the underlying genetic code used to create an organism [Fogel, 1993] 40
- Gonzo** - a GA visualization tool designed to support peoples understanding of their GA's search behaviour 177
- Graphic Density** - the number of marks used to construct a graphic. Bertin proposed that there is an optimum number of marks per square centimeter in any given figure. Although the optimum number was considered to vary with the number of different marks being used, the implantations being applied, the retinal variables being employed and the observer's reading habits, a maximum graphic density of 10 signs per cm^2 was recommended 141
- Hamming Distance** - The number of different bits between two binary strings e.g. the Hamming distance from 1001010101 to 1001000111 is 2 72
- Henson** - an extendable GA visualization framework 160
- Image Component Length** - the number of parts that define a component of information, e.g. the number of values of a variable 134
- Implantation** - a type of mark used in a sign i.e. a point, line or area 135
- Imposition** - the use of the two planar dimensions in an image 135
- Information** - the content of a thought [Bertin, 1983] 134
- Information Components** - a finite set of variational concepts that make up an item of information 134

- Information Elements** - the parts that define a component of information, e.g. the values of a variable 134
- Information Invariant** - an invariant concept that relates to the subject of an item of information . . . 134
- Loci** - plural of Locus 41
- Locus** - the position or site of a gene, mutation, etc. on a chromosome 41
- Maximum Integer Problem** - a GA test problem that attempts to maximize the integer value of the chromosomes in the GA's population 196
- Monosemic Communication** - signs with a single meaning e.g. mathematics and graphics 133
- Offline Fitness** - the mean current-best fitness rating across all generations [De Jong, 1980] 86
- Online Fitness** - the mean fitness rating of all of the chromosomes in the population [De Jong, 1980] . . 86
- Ordered Level of Organization** - a level of organization that includes all the concepts that permit a ranking of elements in a universally acknowledged manner e.g. "this is more than that and less than the other" 134
- Ordered Perception** - a variable is ordered when it is not necessary to consult the legend to be able to order the categories i.e. the order of the signs is universal and immediately perceptible. Texture, value, size and the two planar dimensions, are ordered visual variables 136
- Panasemic Communication** - signs with undefined or subjective meaning, e.g. music or subjective imagery 133
- Pay-Off Matrix** - a matrix used to define the scores attributed by an evolutionary program to a pair of predicted output and actual output values for a finite state machine 44
- Phenotype** - the manner of response contained in the physiology, morphology and behaviour of an organism [Fogel, 1993] 40
- Polysemic Communication** - signs with multiple meanings e.g. language and figurative imagery . . . 133
- Population** - a collection of items under consideration e.g. a set chromosomes 41
- Qualitative Level of Organization** - a level of organization that includes all the concepts of simple differentiation and involves two perceptual approaches; association - "is this similar to that," and selection - "is this different to that" 134

- Quantitative Level of Organization** - a level of organization that is attained when a countable unit is used e.g. “this is a quarter of that and four times the other” 134
- Quantitative Perception** - a variable is quantitative if an observer is be able to perceive the numerical ratio between the signs without consulting the legend. A simple test for quantitative perception is to ask an observer what the value of the largest sign would be if the smallest sign's value was one. Size is the only quantitative retinal variable 136
- Royal Road Problem** - a GA test problem that rewards binary chromosomes for the number of building blocks they contain. For example, a 64 bit binary chromosome is split into eight 8 bit sections: for each 8 bit section containing a string of eight 1's, 8 points are added to the chromosome's fitness rating. For each of the four 16 bit sections containing a string of sixteen 1's, sixteen points are added to the chromosome's fitness rating. For each of the two 32 bit sections containing a string of thirty two 1's, 32 points are added to the chromosome's fitness rating. Finally for a 64 bit chromosome containing all 1s, 64 points are added to the chromosome's fitness rating. The maximum possible fitness rating in this example would be 256 (i.e. $(8 \times 8) + (4 \times 16) + (2 \times 32) + (1 \times 64)$) 200
- Sammon Mapping** - a technique for mapping high dimensional data to fewer dimensions whilst preserving the Euclidean distances between the data points [Sammon, 1969] 146
- Search Space Matrices (“SSM”)** - the use of extensive repartitions in order to construct low dimensional representations of a GA's high dimensional search space [Collins, 1996], [Collins, 1997] and [Collins, 1998] 147
- Selective Perception** - if the eye can isolate all the elements with a specific visual variation to form groups or families within an image and disregard all the others, this variation is selective. Size, value, texture and colour are associative visual variables, as is orientation for point and line implantations (but not areas) 136
- Single Point Crossover** - a typical crossover operator that splits two parent chromosomes at a random point along each chromosome and crosses (i.e. swaps) the two parts of each parent chromosome to produce two offspring 48
- Software Visualization (“SV”)** - the use of the crafts of typography, graphic design, animation and

cinematography with modern human-computer interaction technology to facilitate the human understanding and effective use of computer software [Price et al., 1993] 35

Steady State Reproduction - a type of reproduction sometimes used in evolutionary algorithms that select individual chromosomes for reproduction and (generally) replace the worst individuals with the produced offspring rather than replacing the entire previous generation with a complete new set of solutions 48

Usability-Expressiveness Trade Off - A trade off associated between the usability of a programming environment and the environment's expressive power. This relationship was first identified in [Bell et al., 1991] 130

Visual Variables - the components of the graphic sign system: two planar dimensions, size, value texture, colour, orientation and shape 134

Bibliography

- [Ahlberg and Shneiderman, 1994] Ahlberg, C. and Shneiderman, B. (1994). The Alphaslider: A compact and rapid selector. In Adelson, B., Dumais, S., and Olson, J., editors, *The Proceedings of the International Conference on Human Factors in Computing Systems CHI'94*, pages 365–371, Boston Massachusetts, USA. ACM Press.
- [Alander, 1995] Alander, J. (1995). Indexed bibliography of Genetic Programming. Report 94-1-GP, University of Vaasa, Department of Information Technology and Production Economics. Also available from <ftp://uvasa.fi/cs/report94-1/gaGPbib.ps.Z>.
- [Anderson, 1953] Anderson, R. (1953). Recent advances in finding best operating conditions. *Journal of American Statistical Association*, 48:789–798.
- [Ashby, 1960] Ashby, W. (1960). *Design for a brain*. Wiley, NY., 2nd edition.
- [Baeck, 1992] Baeck, T. (1992). Self-adaption in genetic algorithms. In *The Proceedings of the First European Conference on Artificial Life*, pages 263–271. The MIT press, Cambridge, MA. Also available from http://ls-11www.informatik.uni-dortmund.de/people/baeck/ea_general.html.
- [Baeck, 1996] Baeck, T. (1996). *Evolutionary Algorithms in Theory and Practice (Evolution Strategies, Evolutionary Programming, Genetic Algorithms)*. Oxford University Press, NY.
- [Baeck et al., 1997] Baeck, T., Fogel, D., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. Institute of Physics Publishing.
- [Baeck and Hoffmeister, 1994] Baeck, T. and Hoffmeister, F. (1994). Basic aspects of evolutionary strategies. *Statistics and Computing*, 4:51–63.

- [Baecker, 1981] Baecker, R. (1981). Sorting out sorting. Narated colour videotape. excerpted in ACM SIGGRAPH Video Review 7, 1983.
- [Baecker and Buxton, 1987] Baecker, R. and Buxton, W. (1987). *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan Kaufmann.
- [Baker, 1985] Baker, J. (1985). Adaptive selection methods for genetic algorithms. In *The Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 101–111. Lawrence Erlbaum Associates.
- [Beasley et al., 1993] Beasley, D., Bull, D., and Martin, R. (1993). An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69.
- [Bell et al., 1991] Bell, B., Rieman, J., and Lewis, C. (1991). Usability testing of a graphical programming system: Things we missed in a programming walkthrough. In *The Proceedings of the International Conference on Human Factors in Computing Systems CHI'91*, pages 7–12, New Orleans, LA.
- [Bertin, 1981] Bertin, J. (1981). *Graphics and Graphic Information Processing*. Walter de Gruyter & Co., Berlin and New York.
- [Bertin, 1983] Bertin, J. (1983). *Semiology of Graphics*. The University of Wisconsin Press, Wisconsin.
- [Bishop, 1995] Bishop, C. (1995). EM optimisation of latent-variable density models. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA.
- [Brachman and Anand, 1996] Brachman, R. and Anand, T. (1996). The process of knowledge discovery in databases: A human-centred approach. In *Advances in Knowledge Discovery and Data Mining*, pages 37–58. MIT Press, Cambridge, MA.
- [Brayshaw, 1990] Brayshaw, M. (1990). Visual models of PARLOG execution. Technical Report 64, The Human Cognition Research Group, The Open University, Milton Keynes, UK.

- [Brayshaw, 1994] Brayshaw, M. (1994). *Information Management and Visualziation for Debugging logic Programs*. PhD thesis, The Human Cognition Research Laboratory, The Open University, Milton Keynes, UK.
- [Bremermann, 1962] Bremermann, H. (1962). Optimization through evolution and recombination. In Yovits, M., Jacobi, G., and Goldstein, D., editors, *Self-organizing systems*, pages 93–106. Spartan, Washington D.C.
- [Brindle, 1981] Brindle, A. (1981). *Genetic Algorithms for Function Optimization*. PhD thesis, University of Alberta, Edmonton, Australia.
- [Brown, 1987] Brown, M. (1987). *Algorithm Animation*. MIT Press, Cambridge, MA.
- [Brown, 1988] Brown, M. (1988). Exploring algorithms using balsa-II. *IEEE Computer*, pages 14–36. May 1988.
- [Brown, 1991] Brown, M. (1991). Zeus: A system for algorithm animation and multi-view editing. In *The Proceedings of the IEEE Annual Workshop on Visual Languages*, pages 4–9, Kobe, Japan.
- [Brown and Hershberger, 1992] Brown, M. and Hershberger, J. (1992). Color and sound in algorithm animation. *IEEE Computer*, pages 52–63. December 1992.
- [Brown and Najork, 1993] Brown, M. and Najork, M. (1993). Animating algorithms using 3D interactive graphics. Technical Report 110a, DEC Systems Research Center.
- [Brown and Sedgewick, 1985] Brown, M. and Sedgewick, R. (1985). Techniques for algorithm animation. *IEEE Software*, pages 28–39. January 1985.
- [Chipperfield et al., 1994] Chipperfield, A., Fleming, P., Pohlheim, H., and Fonseca, C. (1994). Genetic algorithm toolbox for use with matlab. Technical Report 512, University of Sheffield, UK.
- [Collins, 1993] Collins, T. (1993). The visualisation of genetic algorithms. Master’s thesis, Department of Computer Science, De Montfort University, Leicester, UK.
- [Collins, 1995] Collins, T. (1995). The visualization of genetic algorithms - related work. Technical Report KMi-TR-19, The Knowledge Media Institute, The Open University, Milton Keynes MK7 6AA, UK.

- [Collins, 1996] Collins, T. (1996). Genotypic-space mapping: Population visualization for genetic algorithms. Technical Report KMi-TR-39, The Knowledge Media Institute, The Open University, Milton Keynes MK7 6AA, UK.
- [Collins, 1997] Collins, T. (1997). Using software visualization technology to help evolutionary algorithm users validate their solutions. In *The Proceedings of the Seventh International Conference on Genetic Algorithms ICGA '97*, pages 307–314, East Lansing, MI., USA.
- [Collins, 1998] Collins, T. (1998). Understanding evolutionary computing : A hands on approach. In *The Proceedings of the IEEE International Conference on Evolutionary Computation ICEC'98*, pages 564–569.
- [Cox and Bruna, 1995] Cox, R. and Bruna, P. (1995). Supporting the use of external representations in problem solving: The need for flexible learning environments. *The Journal of Artificial Intelligence in Education*, 6(2/3):239–302.
- [Dabs and Schoof, 1995] Dabs, T. and Schoof, J. (1995). A graphical user interface for genetic algorithms. Technical Report 98, Lehrstuhl für Informatik II, University Würzburg, DE.
- [Davies, 1991] Davies, S. (1991). The role of notation and knowledge representation in the determination of programming strategy: A framework for integrating models of programming behaviour. *Cognitive Science*, 15:547–572.
- [Davis, 1991] Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, NY.
- [Dawkins, 1986] Dawkins, R. (1986). *The Blind Watchmaker*. Longman.
- [De Jong, 1975] De Jong, K. (1975). An analysis of the behavior of a class of genetic adaptive systems. *Dissertation Abstracts International*, 36(10)(5140B). Doctoral dissertation, University of Michigan, MI. University Microfilms No. 756-9381.
- [De Jong, 1980] De Jong, K. (1980). Adaptive systems design. *IEEE Transactions on Systems, Man and Cybernetics*, pages 566–574.

- [Domingue et al., 1993] Domingue, J., Eisenstadt, M., and Price, B. (1993). VITAL A methodology-based workbench for KBS life cycle support. Technical Report OU/DD342/D1.0, Human Cognition Research Laboratory, The Open University, UK.
- [Domingue and Mulholland, 1997a] Domingue, J. and Mulholland, P. (1997a). The internet software visualization laboratory. In *The Proceedings of the Psychology of Programming Interest Group 9th Annual Workshop*, Sheffield Hallam University, Sheffield, UK.
- [Domingue and Mulholland, 1997b] Domingue, J. and Mulholland, P. (1997b). Staging software visualizations on the web. In *The Proceedings of the IEEE Symposium on Visual Languages VL'97*, Europa Palace Hotel, Capri, Italy.
- [Domingue et al., 1992] Domingue, J., Price, B., and Eisenstadt, M. (1992). A framework for describing and implementing software visualisation systems. In *The Proceedings of the Graphics Interface Conference GI'92*, Vancouver, Canada.
- [Dybowski et al., 1996] Dybowski, R., Collins, T., and Weller, P. (1996). Visualization of binary string convergence by sammon mapping. In Fogel, L., Angeline, P., and Baeck, T., editors, *The Proceedings of the Fifth Annual Conference on Evolutionary Programming EP'96*, pages 377–383, San Diego, CA. MIT Press.
- [Eisenstadt and Brayshaw, 1987] Eisenstadt, M. and Brayshaw, M. (1987). The transparent prolog machine (TPM): An execution model and graphical debugger for logic programming. Technical Report 21a, Human Cognition Research Laboratory, The Open University, UK.
- [Eisenstadt and Brayshaw, 1988] Eisenstadt, M. and Brayshaw, M. (1988). AORTA diagrams as an aid to visualising the execution of prolog programs. Technical Report 29, Human Cognition Research Laboratory, The Open University, UK.
- [Eisenstadt et al., 1990] Eisenstadt, M., Domingue, J., Rajan, T., and Motta, E. (1990). Visual knowledge engineering. *IEEE Transactions on Software Engineering*, 16(10):1164–1177.
- [Fogel, 1993] Fogel, D. (1993). On the philosophical differences between evolutionary algorithms and genetic algorithms. In Fogel, D. and Atmar, W., editors, *The Proceedings of the Second*

- Annual Conference on Evolutionary Programming EP'93*, pages 23–29, La Jolla, CA. Evolutionary Programming Society.
- [Fogel, 1994] Fogel, D. (1994). Evolutionary programming: an introduction and some current directions. *Statistics and Computing*, 4(2):113–129.
- [Fogel, 1998a] Fogel, D. (1998a). *Evolutionary Computation: The Fossil Record*. IEEE Press, NY.
- [Fogel and Ghozeil, 1997] Fogel, D. and Ghozeil, A. (1997). A note on representations and operators. *IEEE Transactions on Evolutionary Computation*, 1(2).
- [Fogel, 1997] Fogel, L. (1997). A retrospective view and outlook on evolutionary algorithms. In *Computational Intelligence: Theory and Applications, 5th Fuzzy Days*, pages 337–342. Springer-Verlag, Berlin.
- [Fogel, 1998b] Fogel, L. (1998b). The valued state space approach and evolutionary computation for problem solving. In *Computational Intelligence: A Dynamic System Perspective*, pages 129–136. IEEE Press, NY.
- [Fogel et al., 1966] Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley and Sons Inc.
- [Fraser, 1957] Fraser, A. (1957). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10:484–499.
- [Friedberg, 1958] Friedberg, R. (1958). A learning machine: part I. *IBM Journal*, pages 2–13.
- [Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley.
- [Goldberg and Deb, 1991] Goldberg, D. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G., editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann.
- [Green, 1982] Green, T. (1982). Pictures of programs and other processes, or how to do things with lines. *Behaviour and Information Technology*, 1(1):3–36.

- [Grefenstette, 1984] Grefenstette, J. (1984). GENESIS: A system for using genetic search procedures. In *The Proceedings of the 1984 Conference on Intelligent Systems and machines*, pages 161–165.
- [Grefenstette et al., 1997] Grefenstette, J. J., Burke, D. S., DeJong, K. A., Ramsey, C. L., and Wu, A. S. (1997). An evolutionary computation model of emerging virus diseases. Technical Report AIC-97-030, Navy Center for Applied Research in Artificial Intelligence.
- [Hand, 1994] Hand, D. (1994). Special issue on evolutionary programming. *Statistics and Computing*, 4(2):47–159.
- [Harvey and Thompson, 1996] Harvey, I. and Thompson, A. (1996). Through the labyrinth evolution finds a way: A silicon ridge. In *The Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware ICES'96*. Springer-Verlag. Also available as ftp://ftp.cogs.sussex.ac.uk/pub/users/inmanh/sil_ridge.ps.gz.
- [Hawkins and Allen, 1991] Hawkins, J. and Allen, R., editors (1991). *The Oxford Encyclopedic English Dictionary*. Clarendon Press, Oxford, UK.
- [Hoffmeister and Baeck, 1992] Hoffmeister, F. and Baeck, T. (1992). Genetic self-learning. In *The Proceedings of the First European Conference on Artificial Life*, pages 227–235. The MIT Press, Cambridge, MA.
- [Holland, 1970] Holland, J. (1970). Outline for a logical theory of adaptive systems. In Burks, A., editor, *Essays on Cellular Automata*. University of Illinois Press.
- [Holland, 1975] Holland, J. (1975). *Adaption in Natural and Artificial Systems*. University of Michigan, 3rd edition.
- [Jones and Forrest, 1995] Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *The Proceedings of the Sixth International Conference on Genetic Algorithms ICGA '95*, pages 184–192, San Francisco, CA. Morgan Kaufmann.
- [Kapsalis et al., 1993] Kapsalis, A., Rayward-Smith, V., and Smith, G. (1993). Fast sequential and parallel implementation of genetic algorithms using the GAMETER toolkit. In *The Proceedings of the International Conference on Neural Networks and Genetic Algorithms*, Innsbruck. Springer-Verlag.

- [Kapsalis and Smith, 1992] Kapsalis, A. and Smith, G. (1992). *The GAMETER Development Toolkit User Interface Manual*. University of East Anglia, Norwich, UK.
- [Kohonen, 1989] Kohonen, T. (1989). *Self-organization and associative memory*. Springer-Verlag, Berlin, 3rd edition.
- [Kolodner, 1993] Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, CA.
- [Koza, 1992] Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection and Genetics*. The MIT Press.
- [Labaw, 1980] Labaw, P. (1980). *Advanced Questionnaire Design*. Abt Books, MA.
- [Larkin, 1989] Larkin, J. (1989). Display based problem solving. In *Complex Information Processing: The Impact of Hernert Simon*. Erlbaum, Hillsdale, NJ.
- [Larkin and Simon, 1987] Larkin, J. and Simon, H. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99.
- [Lawrence et al., 1994] Lawrence, A., Badre, A., and Stasko, J. (1994). Empirically evaluating the use of animations to teach algorithms. Technical Report GIT-GVU-94-07, Graphics Visualization and Usability Center, Georgia Institute of Technology, Atlanta GA.
- [Lewontin, 1974] Lewontin, R. (1974). *The Genetic Basis of Evolutionary Change*. Columbia University Press, NY.
- [Lichtfuss, 1965] Lichtfuss, H. (1965). Evolution eines rohrkrummer diplomarbeit. Technical report, Technische Universtat Berlin.
- [Lieberman, 1984] Lieberman, H. (1984). Steps toward better debugging tools for lisp. In *The Proceedings of the ACM Symposium on Lisp and Functional Programming*, Austin, Texas.
- [Lieberman and Fry, 1995] Lieberman, H. and Fry, C. (1995). Bridging the gulf between code and behaviour in programming. In *The Proceedings of the International Conference on Human Factors in Computing Systems CHI'95*.
- [Maher and de Silva Garza, 1997] Maher, M. and de Silva Garza, A. G. (1997). Case-based reasoning in design. *IEEE Expert*, 12(2):34–41.

- [Mann, 1994] Mann, J. (1994). *X-GAMETER v1.5 User Manual*.
- [McCormick et al., 1987] McCormick, B., DeFanti, T., and Brown, M. (1987). Visualization in scientific computing. *Computer Graphics*, 21(6):1–14.
- [Michalewicz, 1991] Michalewicz, Z. (1991). GENOCOP (GENetic algorithm for NUMerical OPTimization for CONstrained Problems): A genetic algorithm system for optimizing functions with linear constraints. Available from <http://www.coe.uncc.edu/~zbyszek/evol-systems.html>.
- [Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 3rd edition.
- [Mihalisin et al., 1991] Mihalisin, T., Timlin, J., and Schwegler, J. (1991). Visualization and analysis of multi-variate data: A technique for all fields. In *The Proceedings of the IEEE Conference on Visualization Visualization'91*, pages 171–178.
- [Mitchell, 1996] Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA.
- [Mitchell et al., 1991] Mitchell, M., Forrest, S., and Holland, J. (1991). The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Towards a Practice of Autonomous Systems: The Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA.
- [Moher, 1988] Moher, T. (1988). PROVIDE: A process visualisation and debugging environment. *IEEE Transactions on Software Engineering*, 14(6):849–857.
- [Nassersharif et al., 1994] Nassersharif, B., Ence, D., and Au, M. (1994). Visualisation of evolution of genetic algorithms. In *The Proceedings of the World Congress on Neural Networks*, volume 1, pages 1–560–1–565, San Diego, CA., USA.
- [Nelson, 1993] Nelson, G. (1993). Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In *The Proceedings of the Fourth Biennial Art and Technology Symposium*, pages 155–169, Connecticut, USA.

- [Norman and Draper, 1986] Norman, D. and Draper, S., editors (1986). *User Centred Systems Design*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- [Osada et al., 1993] Osada, M., Liao, H., and Shneiderman, B. (1993). Alphaslider: Searching textual lists with sliders. In *The Proceedings of the Ninth Annual Japanese Conference on Human Interfaces*.
- [Pawlowsky, 1995] Pawlowsky, M. (1995). Crossover operators. In Chambers, L., editor, *Practical Handbook of Genetic Algorithms*, volume 1, pages 101–114. CRC Press Inc.
- [Petre et al., 1998] Petre, M., Blackwell, A., and Green, T. (1998). *Software Visualization: Programming as a Multimedia Experience*, chapter 5 - Cognitive Questions in Software Visualization. MIT Press.
- [Pohlheim, 1998] Pohlheim, H. (1998). *Development and Engineering Applications of Evolutionary Algorithms*. PhD thesis, Faculty of Informatics for Automation, Technical University Ilmenau, Germany.
- [Preece et al., 1990] Preece, J., Rodgers, Y., Benyon, D., Davies, G., and Keller, L. (1990). *A Guide to Usability*. The Open University.
- [Preece et al., 1994] Preece, J., Rodgers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-Computer Interaction*. Addison Wesley.
- [Price et al., 1993] Price, B., Baecker, R., and Small, I. (1993). A principled taxonomy of software visualisation. *Journal of Visual Languages and Computing*, 4:211–266.
- [Radcliffe, 1992] Radcliffe, N. (1992). Non-linear genetic representations. In *The Proceedings of the Second Annual Conference on Parallel Problem Solving From Nature*. Elsevier Science Publishers.
- [Rechenberg, 1965] Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Technical Report Library Translation number 1122, Royal Aircraft Establishment, Farnborough, Hants., UK.
- [Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Frommann-Holzborg, Stuttgart.

- [Reiss, 1990] Reiss, S. (1990). Interacting with the FIELD environment. *Software-Practice and Experience*, 20(S1):89–115.
- [Repenning and Ambach, 1996] Repenning, A. and Ambach, J. (1996). Tactile programming: A unified manipulation paradigm supporting program comprehension, composition and sharing. In *The Proceedings of the 1996 IEEE Symposium of Visual Languages*, pages 102–109.
- [Repenning and Ioannidou, 1997] Repenning, A. and Ioannidou, A. (1997). Behavioural processors: Layers between end-users and java virtual machines. In *The Proceedings of the 1997 IEEE Symposium of Visual Languages*, pages 402–409.
- [Ribeiro et al., 1994] Ribeiro, J., Alippi, C., and Treleaven, P. (1994). Genetic algorithm programming environments. *IEEE Computer*, 27(6):28–43. Also available from <ftp://bells.cs.ucl.ac.uk/papagena/game/docs/gasurvey.ps>.
- [Rizki et al., 1993] Rizki, M., Tamburino, L., and Zmuda, M. (1993). Evolving multi-resolution feature detectors. In *The Proceedings of the Second Annual Conference on Evolutionary Programming*, pages 108–118. Evolutionary Programming Society, La Jolla, CA.
- [Roman and Cox, 1993] Roman, G. and Cox, K. (1993). A taxonomy of program visualisation systems. *IEEE Computer*, 26(12):11–24.
- [Roman et al., 1992] Roman, G., Cox, K., Wilcox, D., and Plun, J. (1992). Pavane: A system for declarative visualization of concurrent computations. *Journal of Visual Languages and Computing*, 3:161–193.
- [Ronald, 1995] Ronald, S. (1995). *Genetic Algorithms and Permutation-Encoded Problems; Diversity Preservation and a Study of Multi-Modality*. PhD thesis, University of South Australia.
- [Ronald, 1997] Ronald, S. (1997). Distance functions for order-based encodings. In *The Proceedings of the 1997 IEEE International Conference on Evolutionary Computation ICEC'97*, pages 43–48.
- [Ronald, 1998] Ronald, S. (1998). More distance functions for order-based encodings. In *The Proceedings of the 1998 IEEE International Conference on Evolutionary Computation ICEC'98*, pages 558–563.

- [Ross and Corne, 1994] Ross, P. and Corne, D. (1994). Applications of genetic algorithms. *Journal for the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB)*, 89:23–30.
- [Routen and Collins, 1993] Routen, T. and Collins, T. (1993). Visualisation of A.I. techniques. In *The Proceedings of the International Conference on Computer Graphics and Visualization COM-PUGRAPH'93*, Portugal. ACM Press.
- [Rumelhart and McClelland, 1986] Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1. MIT Press, Cambridge, MA.
- [Sammon, 1969] Sammon, J. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–408.
- [Scaife and Rodgers, 1996] Scaife, M. and Rodgers, Y. (1996). External cognition: How do graphical representations work? *International Journal of Human-Computer Studies*, 45:185–213.
- [Schwefel, 1995] Schwefel, H. (1995). *Evolution and Optimum Seeking*. John Wiley and Sons, Inc. NY.
- [Schwefel and Kursawe, 1998] Schwefel, H. and Kursawe, F. (1998). On natural life's tricks to survive and evolve. In *The Proceedings of the IEEE International Conference on Evolutionary Computation ICEC'98*, pages 1–8.
- [Schwefel, 1968] Schwefel, H. P. (1968). Experimentelle Optimierung einer Zweiphasendüse Teil I. Technical Report No. 35 of the Project MHD–Staustahlrohr 11.034/68, AEG Research Institute, Berlin.
- [Schwefel, 1975] Schwefel, H. P. (1975). *Evolutionasstrategie und numerische optimierung*. PhD thesis, Technical University of Berlin, Department of Process Engineering.
- [Shine and Eick, 1997] Shine, W. and Eick, C. (1997). Visualizing the evolution of genetic algorithm search processes. In *The Proceedings of the 1997 IEEE International Conference on Evolutionary Computation ICEC'97*, pages 367–372.

- [Shneiderman, 1994] Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77.
- [Shneiderman, 1998] Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley Longman, 3rd edition.
- [Spears, 1994] Spears, W. (1994). Visualizing genetic algorithms. Technical Report AIC-94-055, AI Center, Naval Research Laboratory, Washington, D.C.
- [Stasko, 1989] Stasko, J. (1989). *TANGO: A Framework and System for Algorithm Animation*. PhD thesis, Brown University, Providence, RI.
- [Stasko, 1990] Stasko, J. (1990). The path-transition paradigm: A practical methodology for adding animation to program interfaces. *Journal of Visual Languages and Computing*, 1(3):213–236.
- [Stasko, 1994] Stasko, J. (1994). *POLKA Animation Designer's Package. Release 1.15*.
- [Stasko, 1995] Stasko, J. (1995). The PARADE environment for visualising parallel program executions: A progress report. Technical Report GIT-GUV-95-03, Graphics Visualisation and Usability Center, Georgia Institute of Technology, Atlanta GA.
- [Stasko et al., 1993] Stasko, J., Badre, A., and Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. In *INTERCHI '93*, pages 61–66.
- [Stasko et al., 1998] Stasko, J., Domingue, J., Brown, M., and Price, B., editors (1998). *Software Visualization: Programming as a Multimedia Experience*. MIT Press.
- [Stasko and Kraemer, 1992] Stasko, J. and Kraemer, E. (1992). A methodology for building application-specific visualisations of parallel programs. Technical Report GIT-GVU-92-10, Graphics Visualisation and Usability Center, Georgia Institute of Technology, Atlanta GA.
- [Stasko, 1996] Stasko, J. T. (1996). Using student-built algorithm animations as learning aids. Technical Report GIT-GVU-96-19, Graphics Visualisation and Usability Center, Georgia Institute of Technology, Atlanta GA.
- [Syswerda, 1989] Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *The Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA.

- [Thimbleby, 1990] Thimbleby, H. (1990). *User Interface Design*. ACM Press.
- [Todd and Latham, 1992] Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers*. Academic Press.
- [Tufte, 1990] Tufte, E. (1990). *Envisioning Information*. Graphics Press, Cheshire, Connecticut.
- [Tufte, 1983] Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut.
- [Venturini et al., 1997] Venturini, G., Slimane, M., Morin, F., and de Beauville, J. A. (1997). On using interactive genetic algorithms for knowledge discovery in databases. In *The Proceedings of the Seventh International Conference on Genetic Algorithms ICGA '97*, pages 696–703, East Lansing, MI, USA.
- [Wall, 1996] Wall, M. (1996). *GAlib: A C++ Libarray of Genetic Algorithm Components*.
- [Whitley, 1988] Whitley, D. (1988). GENITOR: A different genetic algorithm. In *Proceedings of the Rocky Mountains Conference on Artificial Intelligence*, Denver, Colorado.
- [Williams, 1993] Williams, G. (1993). GECO: A CLOS-based Framework for Prototyping Genetic Algorithms, version 2.0. Further information on GECO can be obtained from george@hsvaic.hv.boeing.com.
- [Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97.
- [Wu et al., 1998] Wu, A. S., Burke, D., De Jong, K., Grefenstette, J., and Ramsey, L. (1998). Visual navigation through a genetic algorithm. Unpublished Report.
- [Wu and Lindsay, 1997] Wu, A. S. and Lindsay, R. K. (1997). A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, 4(2).
- [Wu et al., 1997] Wu, A. S., Lindsay, R. K., and Riolo, R. L. (1997). Empirical observations on the roles of crossover and mutation. In *The Proceedings of the Seventh International Conference on Genetic Algorithms ICGA '97*, pages 362–369.

Appendix A

GA User Questionnaire

The following appendix presents a copy of the original email and questionnaire used in the GA user study described in Chapter 3.

The Visualization of Genetic Algorithms:

Genetic Algorithms typically produce vast quantities of multi-dimensional data on their way toward what is hoped will be a near-optimal solution. Understanding these vast data sets can be a somewhat daunting task. I aim to alleviate this task using Software Visualization techniques. By Software Visualization I mean “the use of the crafts of typography, graphic design, animation and cinematography with modern human-computer interaction technology to facilitate the human understanding and effective use of computer software.”

However, in order to fully realise the potential that Software Visualization offers to GAs it is essential to have a thorough understanding of the tasks and difficulties associated with GAs. In order to gain this insight I need the help of those working with GAs. Therefore, I want to know as much about your experiences with GAs as possible. Please either email me with your anecdotes, or, fill in the attached questionnaire and email it back to me.

The types of things that I am particularly interested in hearing about are;

- - Which problems you have applied GAs to and how successful you found them,
- - What you find difficult about constructing a GA, e.g. designing the evaluation function,

selecting which genetic operators to use, etc.

- - Any problems you may have encountered whilst trying to evaluate a GAs solution(s)?
- - How you would foresee the application of Software Visualization (as defined above) to GAs, e.g. fitness graphs, population analysis, etc.

I have little or no preference as to which form of response I get, please respond by whichever method you feel most comfortable with. Feel free to browse through the attached questionnaire as this may help spark off ideas for any anecdotes you may have. This questionnaire is also available on the World Wide Web as a form document suitable for completion with forms-supporting browsers, such as Netscape and Mosaic. The web page for this questionnaire is <http://kmi.open.ac.uk/trevor/Quest1.html>.

I am relying on the comments and advice that I receive from you, so that I may ensure that this project shall produce something of practical significance. Once a robust version of the GA visualization tool is created it will be made freely available to those of you who have helped in its creation. So please help me to help you.

yours thankfully,

Trevor Collins.

The Visualization of Genetic Algorithms:

The use of filmcraft and animation to illustrate the execution of Genetic Algorithms (“GAs”) in which the User takes on a Director’s role.

Trevor Collins,
Research Student.
The Knowledge Systems Group,
The Knowledge Media Institute,
The Open University.
Walton Hall,
Milton Keynes MK7 6AA, UK.

phone: 01908-654506

email: t.d.collins@open.ac.uk

www: <http://kmi.open.ac.uk/~trevor/trevor.html>

Introduction

In attempting to design a visualization system specifically for supporting the design and application of genetic algorithms, it would be foolish to ignore the ideas and opinions of those involved in that very task. It is for this reason that the following questionnaire has been designed and it is hoped that with your help this will provide some insight into the complex task of GA application.

This is not a performance assessment document, there are no prizes to be won, and there is no hidden agenda, so please be as truthful and informative as you can. Some details on the purpose of the questions asked in the questionnaire are available by clicking [here](#).

Although I do request you to fill in your name and email address this is only so that I may contact you if the need arises. Any information received will be considered private and confidential, your

name shall not be referenced in any associated publications.

If the questions raised within this questionnaire do not apply to your particular use of GAs please do let me know as I do not want to alienate any section of the GA community from using the resulting system. Please feel free to raise any additional issues that you think may be worth exploring. The GA visualization system will be made available to those who have helped in its creation, it is hoped that this will provide some incentive to those who may benefit from its use.

Just in case there is any need to contact you in the future, please type in your

name:

and email address:

Background Information

1. How long have you been using GAs?
2. During this time what have you used GAs for?
3. Why did you use GAs for these tasks?
4. What environment(s) do you use when working with GAs? Please specify each computing environment separately i.e. the computer system, programming language and/or application tool.

Your Approach to GAs

5. What do you find difficult, if anything, about the following set-up steps involved in creating a GA:
 - (a) Defining the mapping between the problem domain and the string representation used by the GA?
 - (b) Producing an effective evaluation function?
 - (c) Choosing the GA's components, e.g. the initial population creation method, what reproduction gene-pool selection criterion to adopt, which genetic operators to apply, etc.?
 - (d) Selecting suitable parameters for the GA, e.g. the population size, the mutation rate (if appropriate), etc.?
 - (e) Are there any other set-up steps that you use before running the GA? If so please note them and any associated difficulties you encounter below.

6. Having applied a GA to a particular problem what approach do you take, in order to:
 - (a) Assess the quality of any solution(s) found?

- (b) Examine how representative the output of the GA is in terms of all the possible points within the problem-space?

What Characteristics to Visualize

The output of a GA typically takes the form of a set of representative strings (“chromosomes”) and their corresponding evaluation ratings (“fitness”). The fitness values are often then illustrated using a fitness versus generation graph. This may show; the highest fitness rating, the average fitness rating, and/or the lowest fitness rating plotted over sequential generations.

This is of course a very useful aid for identifying the relative fitness ratings of the population across different generations, and illustrates one example of the communicative power of graphical representation.

- 7. If the following typical output characteristics were to be represented what advantages or disadvantages, if any, could you foresee?

- (a) All of the individual chromosomes within each population.

Advantages:

Disadvantages:

- (b) A User defined selection of representative chromosomes.

Advantages:

Disadvantages:

- (c) The rate of change in the populations fitness values, i.e. the gradient values of a fitness versus generation graph.

Advantages:

Disadvantages:

- 8. As well as directly illustrating the output of the GA, visualization could be used to represent additional information either derived from the output dataset or recorded separately. If visualization were used to represent the following characteristics what advantages or disadvantages, if any, could you foresee?

- (a) The chromosomes in the reproduction gene-pool.

Advantages:

Disadvantages:

- (b) The occurrence of mutation in chromosomes where a mutation operator has been applied.

Advantages:

Disadvantages:

- (c) The internal actions of the genetic operators being applied to the chromosomes, e.g. the splitting and crossover between two chromosomes by a single point crossover operator.

Advantages:

Disadvantages:

- (d) A “similarity” rating for each chromosome based on how little they differed to the fittest chromosome, e.g. a ten bit binary chromosome that differed from the fittest chromosome in three of its bit positions (“loci”) may have a similarity rating of 0.7.

Advantages:

Disadvantages:

9. Please specify any other direct or indirect characteristics that you would be interested in seeing visualized.

Interaction Opportunities

The most common form of GA interaction is that of set-up and run, i.e. the algorithm and its parameters are defined in a set-up phase, and then executed in a run phase. The resulting output is typically examined after execution with any interesting solutions being further scrutinised at the User's discretion.

Software Visualisation however offers two-way interaction throughout a Genetic Algorithm's execution. This could be applied simply to permit some control over the speed of execution so as to further examine the visual representations for each generation, or in a more direct manner to manipulate the algorithm's parameters or the current generation's internal values.

10. How helpful, or destructive, would you find each of the following interaction opportunities for your use of GAs?
 - (a) Execution control through the use of a control panel to run, pause, step forward, step backward, save a snapshot, and/or stop execution.
 - (b) Editing the algorithm's parameters during execution.
 - (c) Editing the population's chromosomes between two generations.
 - (d) Editing the reproduction gene-pool's chromosomes within a generation.

11. Please specify any other forms of additional interaction that you would consider beneficial.

Any Other Comments

12. Do you have any other suggestions on how GAs could be made easier to use? Or any other comments at all about GAs? Please note them below.

Future Contact

13. Finally, would you have any objection to being contacted in the future with reference to this project and the evaluation of the resulting GA visualization system?

Yes. I would object to being contacted in the future.

No. I would not object to being contacted in the future.

Thank you very much for taking the time to complete this questionnaire. I hope you found it interesting. Providing I received your consent to contact you again, I shall email you once a robust system is available and inform you of the associated anonymous ftp site. If you are happy with your responses please email them back to me - t.d.collins@open.ac.uk

Trevor Collins.

The Knowledge Systems Group,

The Knowledge Media Institute,

The Open University.

Walton Hall,

Milton Keynes MK7 6AA, UK.

Office Phone: +44 908 654506

Departmental Fax: +44 908 653169

Email: t.d.collins@open.ac.uk

WWW: <http://kmi.open.ac.uk/trevor/trevor.html>

Appendix B

GA User Study: Results Summary

This appendix presents a synopsis of the results of the questionnaire described in the Chapter 3. A summary of the responses for each question is given along with quoted extracts taken from the returned questionnaires. Nineteen completed questionnaires were received, the contents of which are included in Appendix C. In order to aid the explanation of the responses received, the respondents were grouped into three categories based on the respondents motivation and interest in GAs:

1. GA Theory Group; those respondents interested primarily in the theory of GAs (3 people).
2. GA Research Group; those respondents concerned in the application of GAs but as a direct result of their interest in GA research (8 people).
3. GA Applications Group; those respondents concerned primarily with solving a problem, for which GAs offer an effective approach (8 people).

For the remainder of this appendix these three groups will be referred to as the “Theory Group,” “Research Group,” and “Applications Group” respectively.

B.1 Background Information

Questions 1, 2 and 3

The first three questions were intended to identify the respondents’ amount of experience with GAs, their area of interest and motivation for using GAs. Tables B.1, B.2 and B.3 include extracts from

Table B.1: Questions 1, 2 and 3. The areas of interest for each respondent in the theory group and the length of time they have been using GAs.

RESPONDENT	REPORTED INTERESTS	EXPERIENCE
<i>T1</i>	“Research in: solving TSP, distributed GA’s and the effect of the underlying topology. Comparative studies of representation spaces. . . Interest in seeing how well GAs work on such problems” (TSP = Traveling Salesperson Problem)	“approx. 2 years”
<i>T2</i>	“Research on representation and role of mutation.”	“3 years”
<i>T3</i>	“Various timetabling and scheduling problems, . . . Primarily because evolutionary algorithms are my principle research interest. For practical problems they promise flexibility and fast prototyping, though not necessarily best results of course; this very point is part of my research, however.”	“About 4 years.”

the responses to questions 1, 2 and 3 from each user group.

The respondents’ amount of time working with GAs varied from two months to seven years and the nature of their of experience, i.e. the problems that they worked on, varied considerably. Only four of the nineteen responses came from people working outside of an academic institution. One of whom (a member of the research group) had carried out the work referred to in his response whilst previously at university. The other three non-academics were all members of the applications group.

Question 4

Users were asked to detail the computer system, programming language and/or application tool they used. Summaries of the responses received from the three groups of respondents are given in Tables B.4, B.5, and B.6. The most common computer systems, used by 16 of the 19 respondents, were UNIX based machines (division between groupings; theory 3/3, research 7/8, and applications 6/8). The second most popular platform was PCs, used by 7 of the 19 respondents (group divisions; theory 2/3, research 2/8, applications 3/8).

The most common programming languages used were C (12/19; theory 2/3, research 4/8, appli-

Table B.2: Questions 1,2 and 3. The areas of interest for each respondent in the research group and the length of time they have been using GAs.

RESPONDENT	REPORTED INTERESTS	EXPERIENCE
<i>R1</i>	“Playing Prisoner’s Dilemma. . . for my project I am working with relating GAs to visual images of creatures, kind of like Todd and Latham’s stuff.”	“about 2 months”
<i>R2</i>	“Research, mainly in neural net construction . . . Seemed like an interesting idea at the time ;-)”	“1 year”
<i>R3</i>	“Optimization of systems . . . My first task was the development of a Genetic Algorithm Toolbox for Matlab.”	“Nearly 2 years.”
<i>R4</i>	“Various optimization problems . . . Designing a GA toolkit, GAmeter . . .”	“approx. 2 years.”
<i>R5</i>	“Standard cell placement. It’s part of the problem of designing silicon chips. . . My interest in GAs comes first. . .”	“2 years”
<i>R6</i>	“Determining the best transmission scheme and data rate for a baseband communications system. Designing FIR filters. Producing bit sequences with special autocorrelation functions. . . It was more a case of selecting tasks that the GA could be applied to - I’m working on a project to investigate the use of GAs in the design of communication systems.”	“2 years.”
<i>R7</i>	“Computer Architecture/Microprocessor Design. . . I confess that I have always found it fascinating and jumped at the justified chance to play with it in my research”	over 2 years
<i>R8</i>	“Algorithm optimization, curve fitting. . . GAs offer a general method for problem solving optimization problems; also because of interest in the GAs themselves”	“4 or 5 years”

Table B.3: Questions 1,2 and 3. The areas of interest for each respondent in the applications group and the length of time they have been using GAs.

RESPONDENT	REPORTED INTERESTS	EXPERIENCE
A1	“Optimization of designs in the mechanical engineering field ... We found out the GA was much more effective at solving large problems.”	“almost a year”
A2	“I am using GAs for the design of Predictive Controllers. ... Because classical methods of optimization cannot solve the problem ...”	“18 months”
A3	“biological applications: aligning protein sequences folding RNA molecules ...”	“two years”
A4	Protein analysis	“2-3 years”
A5	“Optimization, adaptive search to identify design options, integration with NN.”	“approx 3 yrs”
A6	“Process planning. Mechanical design. Mechanical durability assessment test setup procedure ...”	“4 years on and off”
A7	“Studying the optimum structure of the Australian sheep breeding industry.”	“About 4 years”
A8	“A variety of scientific problems.”	“About 7 years”

cations 6/8) and C++ (7/19; theory 2/3, research 3/8, applications 2/8). Only two respondents said they were using GA specific toolkits (both of whom were involved in the development of their respective toolkits). 9 of the 19 users reported having used more than one system, 7 of whom had used more than one language.

Table B.4: Question 4. A summary of the environments used by members of the GA theory group

RESPONDENT	MACHINE			LANGUAGE		
	UNIX	DOS	Macintosh	C	C++	Smalltalk
<i>T3</i>	†	†		†		
<i>T1</i>	†		†	†	†	
<i>T2</i>	†	†			†	†

Table B.5: Question 4. A summary of the environments used by members of the GA research group

RESP.	MACHINE				LANGUAGE				TOOLKIT	
	UNIX	"Various"	PC	DOS	C	C++	"Various"	Matlab	GAMeter	GA toolbox
<i>R5</i>	†				†					
<i>R7</i>	†				†					
<i>R2</i>	†					†				
<i>R6</i>	†					†				
<i>R8</i>		†					†			
<i>R1</i>	†				†	†				
<i>R4</i>	†		†	†	†				†	
<i>R3</i>	†		†					†		†

Table B.6: Question 4. A summary of the environments used by members of the GA applications group

RESP.	MACHINE				LANGUAGE					
	UNIX	DOS	VMS	Amiga	C	C++	Matlab	Pascal	Fortran	Pop11
A2	†	†			†	†				
A4	†				†	†				
A1		†		†	†				†	
A5	†				†					†
A8	†				†					
A3	†		†		†					
A6	†						†			
A7		†						†		

B.2 Your Approach to GAs

Question 5

After identifying the respondents amount of experience, use of GAs, motivation for using GAs, and working environment, the questionnaire tried to identify any difficulties the respondents encountered whilst applying GAs. Users were asked to specify what they found difficult if anything about defining the problem representation (see Tables B.7, B.8 and B.9), defining the evaluation function (see Tables B.10, B.11 and B.12), choosing the components of the algorithm (see Tables B.13, B.14 and B.15), setting the algorithm's parameters (see Tables B.16, B.17 and B.18), and to describe any other set-up steps they use prior to running their GA and any difficulties they associate with them (Table B.19). In this way the respondents working practice and the difficulties they encounter can be identified. The horizontal separation lines used in these and the remaining tables in this chapter are used to group the different types of responses.

Overall the majority of respondents associated some difficulties with defining the problem representation (8/19) and evaluation function (12/19). Those involved primarily in the application of GAs, i.e. those in the research and applications groups, identified more difficulties with defining an effective evaluation function (research 6/8, applications 5/8) than defining the problem representation

Table B.7: Question 5.1. The difficulties found by members of the theory group whilst defining the mapping between the problem domain and the string representation used by the GA.

RESPONDENT	REPORTED DIFFICULTIES
<i>T2</i>	“String representation is limiting. Not useful for all problems. Better representations exist.”
<i>T1</i>	“This is to my mind the most important step in any algorithm, perhaps more important than the choice of algorithm.”
<i>T3</i>	“... I don't find this difficult, so much as I find it a fascinating area for experimentation. Nevertheless, in many senses this is perhaps the most difficult bit ...”

Table B.8: Question 5.1. The difficulties found by members of the research group whilst defining the mapping between the problem domain and the string representation used by the GA.

RESPONDENT	REPORTED DIFFICULTIES
<i>R6</i>	“Nothing - this is generally very straightforward.”
<i>R1</i>	“... Prisoner's Dilemma this is really easy. I imagine that the mapping onto geometric objects will be much harder.”
<i>R3</i>	“No real problem. ... However, I know, that there are a lot of problems, were the mapping/embedding is difficult.”
<i>R7</i>	“For my problem, this is not too much of a problem. For other problems, this is a major issue, ...”
<i>R2</i>	“Only a problem when there are lots of constraints.”
<i>R8</i>	“One of the hardest problems, if not the hardest.”
<i>R5</i>	“That's the problem. That's what makes the use of GAs like the mensa test. ...”
<i>R4</i>	“This is usually the most important stage ... dictate the ease (or lack of) that the following steps will be implemented.”

Table B.9: Question 5.1. The difficulties found by members of the applications group whilst defining the mapping between the problem domain and the string representation used by the GA.

RESPONDENT	REPORTED DIFFICULTIES
<i>A2</i>	“Not difficult”
<i>A1</i>	“... this mapping is no problem at all.”
<i>A4</i>	“... very simple mapping.”
<i>A6</i>	“The initial creation of the population would produce many unfeasible solutions. Development of the representation method has (hopefully) solved this.”
<i>A7</i>	“An early problem was the tendency of many strategies to produce impossible results. ... I finally fixed this by trying to ensure that the genes would produce legal results.”
<i>A3</i>	“... the ‘naive’ approach rarely works. Thus, the mapping seems to be the most crucial point in the strategy of designing a GA.”
<i>A8</i>	“... not necessarily difficult, but clearly important.”

Table B.10: Question 5.2. The difficulties found by members of the theory group whilst defining the evaluation function.

RESPONDENT	REPORTED DIFFICULTIES
<i>T1</i>	“I’m less interested in this – as I generally look at pretty precise TSP problems or whatever, and investigate the landscape and other such things genetic.”
<i>T3</i>	“I don’t tend to find this as crucial as 5.1, ... probably because the design of this usually follows fairly directly from it.”
<i>T2</i>	“Only when there are conflicting criteria.”

Table B.11: Question 5.2. The difficulties found by members of the research group whilst defining the evaluation function.

RESPONDENT	REPORTED DIFFICULTIES
<i>R1</i>	“Again, for PD this is quite easy. . . .”
<i>R4</i>	“This will either drop out of the problem objective or the representation. . . .”
<i>R5</i>	“For this project, yes, that took time. But that was “just programming.” . . .”
<i>R7</i>	“I call a simulator in my objective function. It took me a long time to write this simulator. . . .”
<i>R3</i>	“Here goes the work. 80%-90% of the time for programming/solving the problem is needed for implementing the evaluation function.”
<i>R8</i>	“Hard for combinatorial problems rather than function optimization; often depends on 5.1 in such cases.”
<i>R2</i>	“Difficult when the final fitness is a number of attributes. . . .”
<i>R6</i>	“It is important that there are no weaknesses in the evaluation function definition, . . . Producing effective evaluation functions is most difficult when a trade-off or compromise is required between a number of system performance measures.”

Table B.12: Question 5.2. The difficulties found by members of the applications group whilst defining the evaluation function.

RESPONDENT	REPORTED DIFFICULTIES
<i>A2</i>	“Not difficult”
<i>A3</i>	“In my case, the evaluation function already exists, so most of the time there is no real choice.”
<i>A1</i>	“That can be quite difficult. . . . there are problems in the field of multi-variate fitness evaluation. Furthermore, in the field of mechanical engineering, . . . the importance of these criteria are not as “fixed” as you would like.”
<i>A4</i>	“Hard to describe what a “GOOD” protein is, but thats a problem with the field not with GAs”
<i>A8</i>	“This can be difficult in many scientific problems; . . . 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 behavior of the algorithm.”
<i>A7</i>	“This is by far the biggest job, as theoretical genetics are very complex (for me anyway). . . .”
<i>A6</i>	“Was initially a problem to define one which gave good solutions enough of an advantage over weaker members, but which did not completely exclude these members.”

(research 3/8, applications 3/8). Whilst the reverse was the true for the theory group, 2 of the 3 theory group respondents identified difficulties experienced whilst defining the problem representation and only 1 identified any problems with constructing the evaluation function

The two members of the theory group that identified difficulties with the problem representation noted that choosing an effective problem representation was the most important step of any algorithm. They also considered the evaluation method to be less interesting, using either predefined evaluation methods or evaluation methods that followed directly from their representation.

From the opinions expressed here it would appear that the difficulties people experience whilst defining their problem representations and evaluation functions are dependent on the task being performed as well as the complexity of the problem and the simplicity (or salience) of the representation being used. The task of the theory group respondents was to study the problem solving method rather than to solve a specific problem, where as the respondents in the research and applications groups were specifically involved in the task of problem solving. The key problem identified by the responses of both the research group and applications group was the problem of working on an ill-defined or poorly understood problem domain. Respondents working in well-understood problem domains did not encounter the same difficulties representing and evaluating their problem solutions as those working in less well known domains. However, this problem cannot be ignored as EAs are one of the few search algorithms that can be applied effectively to ill-defined problems [Mitchell, 1996, page 156].

Table B.13: Question 5.3. The difficulties found by members of the theory group whilst selecting the suitable algorithm components.

RESPONDENT	REPORTED DIFFICULTIES
<i>T3</i>	“...time spent on other matters tends to be far more fruitful than endless tuning of parameters and components. So I tend to use fixed overall choices for these, ...”
<i>T1</i>	“genetic operator very important and hard to pick. the rest ain't too important in my opinion.”

Fewer difficulties were associated with selecting the EA's components than with defining the problem representation or evaluation function. The majority of respondents re-used algorithm designs

Table B.14: Question 5.3. The difficulties found by members of the research group whilst selecting the suitable algorithm components.

RESPONDENT	REPORTED DIFFICULTIES
<i>R3</i>	“If I don’t define special parameters the Toolbox uses default parameter. This includes every part of the algorithm. . . . I can change everything. However, most of the time I don’t have to.”
<i>R4</i>	“. . . most of these components will be apparent from the representation. . . .”
<i>R8</i>	“The operators depend very heavily on the representation. . . .”
<i>R6</i>	“Early experiments produced a reliable structure for the GA which has been applied without any problems to a variety of applications”
<i>R7</i>	“I read the literature and integrated what I learned. I admit this may not be optimal, and knowing what is optimal would be good.”
<i>R2</i>	“Creating initial population can sometimes be time-consuming when there are a number of constraints. . . .”
<i>R5</i>	“Difficult isn’t the word. . . my whole approach is based on a special kind of population seeding! i.e. population creation. As for the other GA components you mention, really I’ve stuck to somebody else’s published details about an algorithm, which I am trying to improve upon.”
<i>R1</i>	“. . . Selecting parents is where I have a lot of problems . . .”

Table B.15: Question 5.3. The difficulties found by members of the applications group whilst selecting the suitable algorithm components.

RESPONDENT	REPORTED DIFFICULTIES
<i>A2</i>	“Not difficult but complicated”
<i>A6</i>	“Choice not really a problem. The parameters used with them, however, make a large difference to results and solution time.”
<i>A1</i>	“... These can be selected at runtime and turned on and off while the algorithm is running. This gives the user the ability to experiment with the different methods and to gain more insight into them.”
<i>A7</i> <i>A3</i>	“I worked through Goldberg’s book, ..., then modified the programs taking into account his comments on potential improvements and any ideas that came to me at the time, ...” “... I am now working with a model using most of the features described by Davis in ‘The handbook of GAs’. ... It seems to me much more worth spending time on the quality of the mapping and the quality of the operators.”
<i>A8</i>	“A bit of trial and error is often required. One becomes more familiar with certain strategies, and I suppose one tends to favour those strategies, perhaps unreasonably, over others. ...”

that had been effective in the past, 2 could interactively change their algorithm's components and parameters during evolution. Overall only 4 of the 19 respondents identified any difficulties selecting their algorithm's components (group divisions; theory 1/3, research 3/8 and applications 0/8). 3 of the 4 respondents (all members of the research group) specifically cited the source of their difficulties. Two of these respondents experienced problems with their initialization operator, one worked on GAs for constructing neural networks, the other worked on cell placement problems for designing chips, both of these problem domains can suffer from the random construction of unfeasible solutions. A third respondent working on evolving strategies for playing the prisoner's dilemma game had experienced difficulties producing an effective selection operator.

Table B.16: Question 5.4. The difficulties found by members of the theory group whilst selecting suitable algorithm parameters.

RESPONDENT	REPORTED DIFFICULTIES
<i>T1</i>	"No opinion"
<i>T3</i>	"...time spent on other matters tends to be far more fruitful than endless tuning of parameters and components. So I tend to use fixed overall choices for these, ..."
<i>T2</i>	"This is a difficult problem, as parameter settings drastically affect the efficiency of the GA."

The fourth and final specified set-up step on which the respondents were asked to comment was the selection of appropriate algorithm parameters (see Tables B.16, B.17 and B.18). This created some difficulties for 10 of the 19 respondents (theory 1/3, research 3/8 and applications 6/8). The source of difficulty, evident in these responses is the invisible link between the algorithm's parameters and its performance. Typically respondents either used trial and error to select appropriate parameter settings or adopted parameter settings recommended in the GA literature. Three of the respondents could interactively change their parameter settings during execution (2 from the research group and 1 from the applications group).

Only two respondents (2/19) carried out any additional steps to those outlined in the generic definition of a GA. Both of these respondents identified setting their algorithms' termination conditions

Table B.17: Question 5.4. The difficulties found by members of the research group whilst selecting suitable algorithm parameters.

RESPONDENT	REPORTED DIFFICULTIES
<i>R3</i>	“If I don't define special parameters the Toolbox uses default parameter. This includes every part of the algorithm. ...I can change everything. However, most of the time I don't have to.”
<i>R4</i>	“I use very basic parameters initially ... , with GAmeter, it is very easy to change parameters at any time. Thus the initial parameter settings does not worry me too much as I know they can be changed at will”
<i>R8</i>	“Not too hard, ...”
<i>R6</i>	“Early experiments produced a reliable structure for the GA which has been applied without any problems to a variety of applications”
<i>R7</i>	“I read the literature and integrated what I learned. I admit this may not be optimal, and knowing what is optimal would be good.”
<i>R2</i>	“As these parameters are so connected to each other, its difficult to fine-tune each one individually.”
<i>R1</i>	“This is the biggest problem for me. Population size I have figured out by experimentation ... Finding a good mutation rate is a nightmare. ...”
<i>R5</i>	“This is a real crusher, this is where your package would save a lot of time. Setting the parameters is an agony for me. ...It would be nice to be able to watch the run and monitor population diversity, and population movement. ...I'd like to see your package built with the idea of users contributing add-on modules. However thoroughly you build the thing, when I use it I am going to want to add more, and I would want to send my modules to some center of cooperation.”

Table B.18: Question 5.4. The difficulties found by members of the applications group whilst selecting suitable algorithm parameters.

RESPONDENT	REPORTED DIFFICULTIES
A3	“It seems to me that the population size is not a real problem. In my experience, GAs are quite robust regarding this parameter. . . .”
A1	“Population size is chosen rather arbitrarily . . . Mutation rate as well as crossover rate can be adjusted at runtime (constantly).”
A8	“Trial and error, starting with parameters which past experience suggests will be productive.”
A7	“I could not find any good guidance here, so I have experimented . . . I do not know of any good methods for selecting these other than trial and error. . . . I have a file of default settings for all the GA settings and those specific to the problem.”
A4	“Hard to find good parameters, did mostly trial and error, still searching for good parameters.”
A2	“complicated”
A6	“The parameters used . . . make a large difference to results and solution time.”
A5	“yes”

Table B.19: Question 5.5. The other set-up steps identified by members of the three user groups.

RESPONDENT	REPORTED DIFFICULTIES
Research Group <i>R6</i> <i>R7</i> <i>R3</i> <i>R4</i> <i>R8</i> <i>R5</i>	<p>“No.”</p> <p>“Nothing special. ... I take great care to make sure everything is debugged and in-order before I run. ...”</p> <p>“Not at the moment ... quite a few problem need a sophisticated pre-processing. This could speed up the optimization considerably.”</p> <p>“No. ... If it is not working, then a re-think of the representation may be required. If it works OKish, then I play around with parameters, possible new operators, to see if there is any improvement, and how much.”</p> <p>“Choosing when to halt the GA is another problem.”</p> <p>“My approach is to use an assisting optimizer to produce a paradigm solution which is partially optimized. Then I produce a population from it ... The aim is to concentrate search on a small part of the solution space which is yet expected to contain global optimum. ...”</p>
Applications Group <i>A2</i> <i>A1</i> <i>A6</i> <i>A7</i>	<p>“You have mentioned everything”</p> <p>“ We did define some preprocessors to make problem definitions easier, but these have nothing to do with the GA. So the answer is: no.”</p> <p>“No GA related ones.”</p> <p>“... I also set an upper limit to the number of generations ...”</p>

as their only additional step. Table B.19 includes extracts from all of the comments made. Only 3 of the respondents identified any other additional steps; 2 used problem specific processing to aid the efficiency of the algorithm, and 1 as an assisting optimizer. However, steps are carried out separately from the use of the GA and are, therefore, not part of the generic GA design task.

Question 6

Users were then asked to identify what assessment measures they took to ensure the quality of the solutions they found (see Tables B.20, B.21, and B.22) and the diversity of the problem-space sampled during the search (see Tables B.23, B.24 and B.25).

Table B.20: Question 6.1. The steps carried out by members of the theory group to verify the quality of their algorithm's solution(s).

RESPONDENT	COMMENTS
<i>T2</i>	"Pick an instance of the problem with a known solution, so that you can verify that it can be found. Then gather statistics on solving the problems over a number of runs."
<i>T1</i>	"Extract loads of data during runs – not very efficient, but I want to know as much as possible."
<i>T3</i>	"Usually there are benchmark results available; if not, then always compare with SA and various kinds of hillclimbing."

Less than half of the respondents (8/19) took any additional steps to explore the quality of their algorithm's solutions other than to examine the chromosomes' fitness ratings (group divisions; theory 3/3, research 2/8 and applications 3/8). The most common forms of quality testing were to compare the results of multiple GA runs, and to compare the results of the GA against the results of alternative approaches (such as simulated annealing).

Even fewer respondents (6/19) explored how representative the output of their GA was in terms of sampling all the possible points within the problem space (group divisions; theory 1/3, research 4/8 and applications 1/8). Of those that did explore their algorithm's coverage of the search space, examining the solutions found during multiple runs was the most common approach.

Table B.21: Question 6.1. The steps carried out by members of the research group to verify the quality of their algorithm's solution(s).

RESPONDENT	COMMENTS
<i>R2</i>	“Just by assessing the fitness functions”
<i>R8</i>	“In most cases the fitness function itself was used; in the curve case, visual inspection was also useful.”
<i>R6</i>	“In some cases the ideal solution is known. The GA has been found to produce close to ideal solutions.”
<i>R3</i>	“Have to look to the data/results. You have to understand the problem, otherwise you can't weight the solution of the GA. . . .”
<i>R5</i>	“I'm just comparing solution quality to that found by my rival's GA.”
<i>R4</i>	“Depending on the problem in question. Try against an exact method . . . Try against a specialized heuristic. try against a general heuristic, SA, TS, etc. . . .”
<i>R7</i>	“Repeatability is a major way for me to know the GA is not simply hacking about. . . I sanity check, and plot the gene and objective values as the simulations proceed. Premature convergence has been the biggest GA problem I have had to address. . . .”

Table B.22: Question 6.1. The steps carried out by members of the applications group to verify the quality of their algorithm's solution(s).

RESPONDENT	COMMENTS
<i>A6</i>	"The evaluation of my problem gives a percentage match. Therefore, a match of 100% is a perfect solution."
<i>A5</i>	"constraint satisfaction"
<i>A1</i>	"In mechanical engineering, it is quite easy to check the results against existing designs."
<i>A3</i>	"We use a benchmark, an exhaustive problem that can provide a guaranteed optimal solution for a small problem."
<i>A2</i>	"1. Extended searching 2. Quality of the final solutions (final performance)"
<i>A8</i>	"Comparison with literature results if available. Comparison with results yielded by conventional approaches on the same data. Statistical analysis of the results yielded by the GA. Comparison between results of repeated runs."
<i>A7</i>	"I normally test the problem 5-10 times, compare it with hill-climbing results and also use my own intuition ... I also look at intermediate calculations used in the optimum to check that they make biological sense."

Table B.23: Question 6.2. The steps carried out by members of the theory group to verify the quality of their algorithm's search path.

RESPONDENT	COMMENTS
<i>T1</i>	"Compare with other algorithms, known bounds etc. This is important to me."

Table B.24: Question 6.2. The steps carried out by members of the research group to verify the quality of their algorithm's search path.

RESPONDENT	COMMENTS
<i>R6</i>	"Haven't bothered."
<i>R4</i>	"...I apply GAs in an industrial context, where the quality of the output is more important than how it fares to all other points in the search space. ..."
<i>R3</i>	"If the evaluation function is smooth, you don't need such a long run of the GA, if it is more chaotic - then you have a problem and you are lost in the GA-space."
<i>R1</i>	"...I compare the values of the genes in each chromosome with the average value for each gene ..."
<i>R8</i>	"...multiple runs were used to see how often the same solutions resulted."
<i>R5</i>	"I'm running the GA repeatedly from different starting points ... comparing GA outputs in terms of fitness, absolute phenotype features, and phenotypic features considered more abstractly."

Table B.25: Question 6.2. The steps carried out by members of the applications group to verify the quality of their algorithm's search path.

RESPONDENT	COMMENTS
<i>A1</i>	“We didn't do extensive research on this, because we got satisfying results, which indicated that the problem space was searched quite well.”
<i>A5</i>	“heuristics, otherwise difficult”
<i>A2</i>	“The main point is the final performance considering at the same time the current practical issues.”
<i>A7</i>	“I rely on my intuition and knowledge of the subject to check any solutions that do not appear to be produced by the GA. . . .”
<i>A8</i>	“Repeated runs. Statistical analysis of runs. Investigation of the surface through gradient search and other local search techniques. Visualization of the surface. Comparison with random search results. Theoretical methods if available.”

B.3 What Characteristics to Visualize

Question 7

Having got an impression of the users background and approach to applying a GA and evaluating its results, the questionnaire moved on to ask more specific questions about what could be visualized. Respondents were asked to note the advantages and disadvantages of viewing all the individual chromosomes in each population (see Tables B.26, B.27 and B.28), a user defined sub-set of the chromosomes in each population (see Tables B.29, B.30 and B.31), and the rate of change in the populations' fitness values (see Tables B.32, B.33 and B.34).

Table B.26: Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the theory group.

RESPONDENT	COMMENTS
<i>T2</i>	"Can do it already"
Advantages	
<i>T1</i>	"none for my purposes"
<i>T3</i>	"You can see whats going on! In some cases of course – like photofit generation, or evolving art, this is necessary anyway."
Disadvantages	
<i>T1</i>	"too much to look at"
<i>T3</i>	"Impossible on realistically large problems, so some form of summarization desperately needed."

Eleven of the nineteen respondents identified some advantages for viewing all the individual chromosomes (group divisions; theory 1/3, research 4/8 and applications 6/8), but 14 of the respondents also identified some disadvantages (theory 2/3, research 6/8 and applications 6/8). The majority of respondents noted that although viewing all the individual chromosomes within each population may give an early insight into what was going on, too much information would be produced and this would be difficult for the user to interpret.

As an alternative to viewing all the individuals the respondents were asked to comment on the

Table B.27: Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R1</i>	“Massive amount of information!”
<i>R3</i>	“If only one generation (the actual one) at once”
<i>R6</i>	“Variations between members could be easily observed. Convergence could also be spotted easily”
<i>R7</i>	“Fun. . . . Could get an early-on sense of whats was happening. . . .”
<i>R2</i>	“If clusters were forming around local minima”
<i>R4</i>	“- This is good for seeing how similar (or not) all members of the population are. - It may suggest ways in which improvements could be made to the GA (for example niching). - It may highlight how the GA has become trapped in a local optimum.”
Disadvantages	
<i>R1</i>	“Too much information for a human to usefully digest.”
<i>R3</i>	“too much information, you don’t have to see every bit of information.”
<i>R6</i>	“Too much information displayed at once could hide useful information.”
<i>R4</i>	“- It can be confusing if you have very large bitstrings - not very informative if the genotype/phenotype map is not straightforward.”
<i>R8</i>	“Far too many for this to be useful. The chromosomes themselves are not very meaningful in some of our work”
<i>R5</i>	“If you can understand it, your problem is too simple.”
<i>R7</i>	“I can imagine “visualizing” real-valued parameters. But how do you visualize other problem-specific genetic representations”
<i>R2</i>	“depends on the population size I suppose. execution speed might be a problem”

Table B.28: Question 7.1. The advantages and disadvantages of visualizing the individual chromosomes in each population, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A3</i>	"none"
<i>A8</i>	"None, unless the population were very small. . . ."
<i>A7</i>	"I did this when first starting simple test function, and it did help to verify that my program was doing the right things."
<i>A6</i>	"Good for investigation as to what the GA is doing."
<i>A2</i>	"Supervisory control to all individuals"
<i>A4</i>	"quick analysis of relationships genes"
<i>A1</i>	"A detailed "report" of the current population, providing a lot of data to those that can "read" it correctly."
<i>A5</i>	"all info"
Disadvantages	
<i>A8</i>	"Too much screen clutter"
<i>A7</i>	"I quickly stopped looking at each individual as it is too confusing and not informative."
<i>A6</i>	"Far too much data for me, since I am looking at an application rather than the GA itself."
<i>A2</i>	"1. For large populations? 2. It's difficult to check all the candidates."
<i>A1</i>	"Might confuse people, and might tempt people to draw the wrong conclusions. I mean, the GA is strongly stochastic, so one must always be careful about drawing conclusions from any particular run."
<i>A5</i>	"too much info, unnecessary"

Table B.29: Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the theory group.

RESPONDENT	COMMENTS
<i>T2</i>	“See above” 7.1 - Can do it already
Advantages <i>T3</i>	“Very flexible, to the extent that the user requirements can be varied.”
Disadvantages <i>T3</i>	“Could be easy to hide what’s really happening.”

advantages and disadvantages of viewing a user defined selection of representative chromosomes (see Tables B.29, B.30 and B.31). In this case 14 of the 19 respondents noted some advantages (group divisions; theory 1/3, research 5/8 and applications 8/8), and 4 of the 19 noted some disadvantages (theory 1/3, research 4/8 and applications 2/8). Although this was seen as less of an information swamp the added danger of making a non-representative selection and disregarding important information was a commonly noted fault.

Finally the respondents were asked to comment on the suitability of visualizing the rate of change in the populations fitness values, such as the gradient of a fitness versus generation graph (see Tables B.32, B.33 and B.34). A fitness versus generation graph is in fact the most common visualization used within the EC community for illustrating an algorithm’s evolutionary search and the vast majority of these users reported using some form of this graph. 14 of the 19 respondents identified some advantages for visualizing the rate of change in the populations fitness values (group divisions; theory 2/3, research 7/8 and applications 5/8) and 2 of the respondents identified some disadvantages (theory 1/3, research 1/8 and applications 0/8).

Question 8

In addition to directly illustrating the typical output data of the GA, users were made aware of four alternative visualizations that required additional information to be recorded or derived from the output data and were asked to comment on them. These included; representing the chromosomes in the reproductive gene-pool (see Tables B.35, B.36 and B.37), the occurrence of mutation (see

Table B.30: Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R6</i>	“Less of an ‘information swamp’.”
<i>R1</i>	“Limit the number of things the user has to examine.”
<i>R4</i>	“reduces disadvantage #1 [see 7.1]”
<i>R5</i>	“‘Establishes conventions’
<i>R3</i>	“necessary, if you know/understand the overall meaning of your data, you want to have a look. For instance, I plot the chromosome of the best individual in every generation over all generations. Thus, I get a meaning of the change of the best individual during the optimization.”
Disadvantages	
<i>R8</i>	“Same” [7.1]
<i>R5</i>	“we do not yet know enough to establish such conventions, yet maybe your package should make a stand and be open to change. At the moment everybody does their own thing here.”
<i>R6</i>	“How does the user define a ‘representative’ set of chromosomes. They may well NOT be representative at many or all points of a particular run.”
<i>R1</i>	“The user may not know what he/she is doing, and could pick a non-representative selection, and miss the interesting things.”
<i>R4</i>	“- How do we know the selection is a fair selection? - Gives another burden to the user to decide. - doesn’t help disadvantage #2” [see 7.1]
<i>R2</i>	“as its user-defined I can’t foresee any problems PROVIDED the user knows what subset of strings he/she wants, and how to specify them”

Table B.31: Question 7.2. The advantages and disadvantages of visualizing a user defined selection of the individual chromosomes in each population, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A2</i>	“Ability for someone to be experimented, so that to choose the best possible representation for the specific problem.”
<i>A6</i>	“Better than 7.1, giving some info on the current generation.”
<i>A1</i>	“Would give even better controlled data. this would definitely be better than 7.1.”
<i>A8</i>	“Better. Less screen clutter”
<i>A3</i>	“if properly done, it could help visualizing the emergence of some niche, and maybe their relations”
<i>A7</i>	“My current system shows the 17 gene values for the best 5 individuals. This gives me some idea how things are going, whether they are converging and which genes are still highly variable.”
<i>A4</i>	“again gives you an overall picture of genes changes”
<i>A5</i>	“helpful”
Disadvantages	
<i>A8</i>	“None”
<i>A1</i>	“Same disadvantages [as 7.1], with the additional risk of accidentally disregarding data that might be important after all.”
<i>A5</i>	“there can be a chance to loose novel chromosome structure.”
<i>A7</i>	“Although the best 5 are always shown i usually only look at the best one (shown in a different colour) and use the current minimum, average and maximum to check how the GA is going.”

Table B.32: Question 7.3. The advantages and disadvantages of visualizing the rate of change in the populations' fitness values, as reported by members of the theory group.

RESPONDENT	COMMENTS
Advantages <i>T2</i> <i>T3</i>	“Can do it already. Shows you what is happening during single runs.” “People tend to like this simply because it shows there's something actually happening. If a ‘when fitness get's here we're fine’ line is on the graph, possible for most problems, then the illusion of understanding the GA's progress is comfortably strong.”
<i>T1</i>	“need more than this; need to know the local structure of fitness changes throughout the population”
Disadvantages <i>T2</i>	“Slow down the run”
<i>T3</i>	“None really, except that there might be much messing around on some problems to translate fitnesses into graphable quantities . . .”

Table B.33: Question 7.3. The advantages (Adv.) and disadvantages (Disadv.) of visualizing the rate of change in the populations' fitness values, as reported by members of the research group.

RESP.	COMMENTS
Adv.	
<i>R8</i>	“Not a lot, because the fitness versus generation graph is quite noisy, and so derivatives would be measuring noise.”
<i>R3</i>	“If you plot the fitness of the population (only best individual and/or mean and/or worst), this plot includes the gradient. Thus, normally it is only really necessary if you have the fitness directly. However, one of them is absolutely necessary.”
<i>R6</i>	“Gives an indication of convergence.”
<i>R7</i>	“Yes since this tells the user s/he may be nearly done, or at least near a plateau. Also, I watch the standard deviation of objective values”
<i>R2</i>	“Good to see if population is stagnating, and might need a boost (e.g. load of mutation, or a few new random strings)”
<i>R5</i>	“Essential. Even I report it.”
<i>R1</i>	“See how quickly the chromosomes converge on a solution, also see how stable populations are.”
<i>R4</i>	“- Standard visualization tool everyone can understand. - show convergence of GA, etc.”
Disadv.	
<i>R2</i>	“can't think of any”
<i>R4</i>	“- Too much emphasis can be placed on the graph without going into any detail as to why that pattern occurred.”
<i>R6</i>	“The fitness vs. generation graph is usually very noisy, particularly with high variations in the fitness function between good and bad members. The gradient of this curve would have to be averaged to produce a useful value, and the averaging needed may well depend on the particular application.”
<i>R1</i>	“only useful in conjunction with the other graphs showing fitness, otherwise for example, you could get graphs which bottom out after n generations, but don't tell you how well the populations were actually scoring, just the fact that they had reached a stable state.”

Table B.34: Question 7.3. The advantages and disadvantages of visualizing the rate of change in the populations' fitness values, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A1</i>	"Not sure about these..."
<i>A5</i>	OK, can give some idea about convergence
<i>A8</i>	"A standard method of following the progress of the calculation. Generally gives a useful idea of how things are going."
<i>A6</i>	"Gives info on how the search is proceeding"
<i>A7</i>	"When I do many overnight runs to test the settings I compare them using a graph of fitness versus generation. This shows how quickly different settings reach good values, and how close they get to the highest possible value. This is useful because some settings make the best gains early, but seem to run out of variation and fail to reach the maximum that slower settings can reach. I have to do this manually in Excel from test files produced during the run."
<i>A4</i>	"Very interesting, good way to actually see if what you plan is actually working"
Disadvantages	
<i>A1</i>	"Not sure about these..."
<i>A8</i>	"One often wants a more detailed understanding of what is happening in the population than this graph can give."
<i>A7</i>	"I do not find this useful for examining the actual results (i.e. in terms of my sheep breeding system) except to get some idea whether further increases might be possible if left for more generations."

Tables B.38, B.39 and B.40), the internal actions of the genetic operators (see, Tables B.41, B.42 and B.43), and a similarity rating for each chromosome based on how little it differed from the fittest chromosome (i.e. the hamming distance, see Tables B.44, B.45 and B.46).

Table B.35: Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the theory group.

RESPONDENT	COMMENTS
Advantages	
<i>T2</i>	“can do it. Only useful to check if GA works correctly, and see the effect of hamming cliffs”
<i>T3</i>	“In one sense, this doesn’t tell you much more than the visualization of the entire pool (or bits of it), provided you already know what the selection pressure roughly is. It is occasionally interesting when a very lowly fit chromo is chosen for parenting, but you already know that will happen every now and then. What’s interesting, though, is when poor parents lead to strong children. I had some success in a program which indicated on screen every example of a crossover yielding a child better than both parents. Among other things, this is good to help users justify the use of a GA to their bosses.”

Visualizing the chromosomes in the reproduction gene-pool was not seen as particularly useful. Seven of the respondents noted some advantages (group divisions; theory 1/3, research 4/8, applications 2/8) and four noted disadvantages (theory 0/8, research 4/8, applications 0/8). Visualizing the occurrence of mutation was also met with some indifference. Five of the nineteen respondents noted some advantages (theory 0/3, research 2/8, applications 3/8) and 3/19 noted some disadvantages (theory 0/8, research 3/8, applications 0/8).

Representing the internal actions of the genetic operators used was seen as being an aid for both education and program debugging (see, Tables B.41, B.42 and B.43). Eight of the respondents noted some advantages (group divisions; theory 0/3, research 6/8, applications 2/8) and 3 out of the 19 noted some disadvantages (theory 0/3, research 3/8, applications 0/8).

Table B.36: Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R8</i>	“See above.” [7.1]
<i>R2</i>	“probably not (a) the easiest thing to do in general, (b) the most useful, but probably is problem-dependent to a large degree”
<i>R6</i>	“For binary strings, it would be possible to determine whether every possible allele existed in the initial population, and how well different alleles propagated.”
<i>R7</i>	“Good, I do this already in a non-fancy semi-automated way so I can watch what design is evolving as it proceeds. My entire GA run might take 2+ weeks so I need to watch for sanity as it proceeds.”
<i>R5</i>	“Could be used to spot convergence”
<i>R4</i>	“- Shows how the genetic operators have been working - Shows which children were automatically discarded.”
<i>R3</i>	“see 8.3”
Disadvantages	
<i>R8</i>	“See above.” [7.1]
<i>R2</i>	“could get tricky!”
<i>R6</i>	“For non-binary strings, the number of possible alleles for each gene is likely to be prohibitively high.”
<i>R1</i>	“Way too confusing - I don’t see how a user could get any useful information out of such a picture. Perhaps I’m wrong. Actually, I guess that in the testing stages, when the population size could be kept small, it would be reasonable.”
<i>R4</i>	“- Is this needed?”
<i>R5</i>	“Your package should spot convergence for us. . . .”
<i>R3</i>	“see 8.3”

Table B.37: Question 8.1. The advantages and disadvantages of visualizing the chromosomes in the reproduction gene-pool, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A7</i>	“This is only useful when I have made changes to the program in this section and need to check that I have not introduced a new bug.”
<i>A8</i>	“Depends upon the problem being tackled. We have found such a visualization useful at times.”
Disadvantages	
<i>A8</i>	“None”
<i>A7</i>	“No need unless the program is not working correctly”
<i>A5</i>	“not much info”

Table B.38: Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the theory group.

RESPONDENT	COMMENTS
Advantages	
<i>T3</i>	“Entertaining”
<i>T2</i>	“the showing the occurrence of mutation by itself is not very useful. Percentage of mutations that are better than its “parent” are useful to show the effectiveness of mutation over the run.”
Disadvantages	
<i>T3</i>	“You need to be very selective, since there are so many.”

Table B.39: Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R2</i>	“dunno”
<i>R5</i>	“None.”
<i>R8</i>	“See above.” [7.1]
<i>R3</i>	“see 8.3”
<i>R6</i>	“This could give us an indication of whether the mutation rate was too high (interfering with the evolution process) or too low (allowing stagnation). It could also indicate the introduction of a previously unencountered allele on the chromosome.”
<i>R4</i>	“- Can highlight whether mutation should be increased or decreased.”
Disadvantages	
<i>R2</i>	“dunno”
<i>R5</i>	“Distraction. User should know where mutation will occur.”
<i>R8</i>	“See above.” [7.1]
<i>R3</i>	“see 8.3”
<i>R6</i>	“None.”
<i>R4</i>	“- A lot of overhead for this information.”

Table B.40: Question 8.2. The advantages and disadvantages of visualizing the occurrence of mutation in chromosomes, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A8</i>	“limited”
<i>A6</i>	“Good for following what has been happening, therefore the understanding of what is going on.”
<i>A3</i>	“allow the user to really get a feeling of what the mutation does, and possibly, what are its limits.”
<i>A7</i>	“Same as 8.1”
Disadvantages	
<i>A5</i>	“not much info”
<i>A8</i>	“Since mutation normally causes little change in the string, there wouldn't be a great deal to show! There should be no value in showing the position of mutation, unless for some reason one biases the position. I can't see this being useful.”
<i>A7</i>	“Same as 8.1”

Table B.41: Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the theory group.

RESPONDENT	COMMENTS
Advantages	
<i>T2</i>	“Can't see any, so have not done it.”
<i>T3</i>	“Entertaining”
Disadvantages	
<i>T3</i>	“Surely gets very dull after a while, so maybe use only as an option for demos, new users, etc.”

Table B.42: Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R8</i>	“See above.” [7.1]
<i>R6</i>	“Possible to observe correct operation of the GA.”
<i>R7</i>	“Fun, good teaching/debugging tool. I am not sure how to digest and exploit this info, however, over 1000s of runs in a real application.”
<i>R2</i>	“probably good for education purposes, but when you think of the number of matings that are going to occur in a typical GA run, then it'd probably be too much to take in. . . .”
<i>R5</i>	“Good for educational purposes. Also, all of your ideas for displays might turn out to be useful for debugging the GA.”
<i>R1</i>	“Would be interesting, I guess!”
<i>R4</i>	“- can highlight when the operator has become defunct or when another operator would be more useful.”
<i>R3</i>	“if I want to know, how the GA works, this would be useful. but see below.”
Disadvantages	
<i>R8</i>	“See above.” [7.1]
<i>R6</i>	“Unnecessary.”
<i>R2</i>	“see above” [Advantages]
<i>R5</i>	“none as long as its optional.”
<i>R4</i>	“as above.” [Disadvantages 8.2]
<i>R3</i>	“this would be much to operator specific. Using a GA I want to solve problems. I don't wanna know how the GA works.”

Table B.43: Question 8.3. The advantages and disadvantages of visualizing the internal actions of the genetic operators, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A7</i>	“Same as 8.1”
<i>A6</i>	“Same as above.” [8.2]
<i>A8</i>	“limited”
Disadvantages	
<i>A7</i>	“Same as 8.1”
<i>A8</i>	“Again this would be of interest in illustrating how the GA works, but I think of little value in helping one monitor the action of the algorithm.”
<i>A5</i>	“will not be meaningful in multidimensional problem situation.”

Table B.44: Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the theory group.

RESPONDENT	COMMENTS
Advantages	
<i>T2</i>	“Only bother in terms of fitness. This would be seen by the average fitness being very close to the maximum fitness.”
<i>T3</i>	“As one of potentially many such diversity measures, this is certainly an important thing to show. Helps much in seeing what’s going on”
Disadvantages	
<i>T3</i>	“Needs more sophistication to be truly useful. Eg, there may be several best-fit chromos, all genotypically or phenotypically distinct.”

Table B.45: Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the research group.

RESPONDENT	COMMENTS
Advantages	
<i>R6</i>	“A better indication of convergence than the gradient of the fitness vs. generation graph.”
<i>R7</i>	“Yes, many.”
<i>R2</i>	“good to show clustering around peaks”
<i>R5</i>	“This justifies your enterprise. If I don’t know this I don’t know what my algorithm is doing (that premise currently true!)”
<i>R1</i>	“Aha ! This is very good. I think it is necessary in populations that tend to stabilize, in order to keep variation going (if that’s what you want) but if you want to solve a problem then it could also be useful to push away from local maxima.”
<i>R3</i>	“Similarity to the fittest sounds like a good idea. (I do an histogram of the differences between all individuals between each other. This gives a meaning of the diversity of the population.”
<i>R4</i>	“- could be useful. . .”
Disadvantages	
<i>R6</i>	“None.”
<i>R8</i>	“Far too simple a measure of similarity.”
<i>R1</i>	“As I said above, I don’t really know how to implement this sort of thing - it sounds like statistics to me. Could be quite computationally expensive, maybe?”
<i>R4</i>	“- but it really depends on the problem. For example, in highly epistatic problems, its not the number of different genes but the actual genes that are different which determines how good or bad that solution is. This information would be useless in this case.”

Table B.46: Question 8.4. The advantages and disadvantages of visualizing a “similarity” rating for each chromosome, as reported by members of the applications group.

RESPONDENT	COMMENTS
Advantages	
<i>A6</i>	“Would allow knowledge of the diversity of the population. This would allow, through experimentation on your problem knowledge as to how the search was likely to proceed.”
<i>A3</i>	“this might be quite useful in helping to identify problems that need a ‘niche’ approach”
<i>A1</i>	“All these statistics would give more insight into the GA, and would therefore be quite nice for educational purposes.”
<i>A8</i>	“We’ve used this type of measure a lot. Useful.”
<i>A7</i>	“There might be some value in a convergence value that summarized the whole population so you could check the rate of convergence and decide when no further gains were likely.”
Disadvantages	
<i>A8</i>	“None”
<i>A5</i>	“that is not representative of binary representation”
<i>A7</i>	“Doing this for individual chromosomes would be confusing.”
<i>A1</i>	“It slows down the GA and therefore (in my particular application) the optimization process, which is a disadvantage . . .”

Finally Question 8 asked the respondents to indicate the advantages and disadvantages of illustrating a similarity rating for each chromosome based on how little they differed from the fittest chromosome (see Tables B.44, B.45 and B.46). Such a measure of population diversity was seen as a very favorable feature, however several respondents noted concern over the measure's efficacy. 13 of the 19 respondents noted advantages (theory 1/3, research 7/8, applications 5/8) and 3 noted disadvantages (theory 0/3, research 0/8, applications 3/8).

Question 9

Question 9 asked the respondents to describe any other views that they would be interested in seeing. All of the comments received from members of the theory, research and application groups are given in Tables B.47, B.48 and B.49, respectively.

Table B.47: Question 9. The additional features of a GA which members of the theory group reported that they would be interested in seeing.

RESPONDENT	COMMENTS
<i>T1</i>	"The ratio of accepted changes to non-accepted, the correlation in fitness values as a function of distance in representation space, the frequency of local fitness peaks,... comparisons with other algorithms."
<i>T3</i>	"Generally, info on a variety of interesting events – eg: every time a new best fit arrives, let's see its parent(s) and the operation which produced it. Let's also see those operations in which very good parent(s) led to terrible children. Good also to see what's occurring in and between niches. Eg: an ongoing measure of how child fitness correlates with parent diversity."

This question enabled the respondents to express what they felt about visualization, the responses to this question alone validated the enterprise of the GA user study. Many more ideas were proposed than expected. Ten of the respondents suggested additional visualizations (theory 2, research 4, applications 4). The suggestions made by the theory respondents were for specific summary visualizations of the GA's data, such as the frequency of local fitness peaks, as well as finer detailed data regarding

Table B.48: Question 9. The additional features of a GA which members of the research group reported that they would be interested in seeing.

RESPONDENT	COMMENTS
<i>R2</i>	“It might be interesting if you could look at a ‘family tree’ of an individual chromosome, and see how the fitness improves.”
<i>R5</i>	“Niches. I mean sort things so that similar ones go together. By the way, I’d like string similarity to be pretty flexible, or at least particularly open to add-ons.”
<i>R3</i>	<p>“At the moment I visualize the following things (every 10 generations or so): - fitness value of best individual in the last 20-40 generation, - chromosome of best individual in the last 30-60 generation, - all chromosomes in the actual generation, - all fitness values in the actual generation, - histogram of diversity of chromosomes in actual generation (first try, needs more work).</p> <p>This is quite enough for a good understanding, what’s going on. (For every system I often include system specific visualizations (dynamic optimization -> results of simulation with best individual for instance.))</p> <p>There are lots of new possibilities, if you can make movies and so on. At the moment, the computing power is far too less to think about an implementation. However, I think there should a lot be done. I will do some thinking as well and when you contact me, we can talk about more ideas.”</p>
<i>R4</i>	<p>“This I am very interested in. As a developer of a toolkit, I am always looking at ways in which the visualisation could be improved. But at the same time I think about the overhead caused by this visualisation. My outlook is visualisation is nice, but not for the sake of speed and general usefulness. (i.e. it’s no point added some functionality if most problems don’t need this information or visual guidance.)”</p>

Table B.49: Question 9. The additional features of a GA which members of the applications group reported that they would be interested in seeing.

RESPONDENT	COMMENTS
A2	“You have not left anything for me to think. The questions mentioned above, definitely have only advantages.”
A1	“In the field of optimization, the independent variables are also displayed (at least, we display them). In general, one could perhaps say that the “data that the chromosomes represent” should be shown (optionally).”
A8	“Varies greatly from one application to the next. The most useful factors we follow relate to the degree of diversity within the population.”
A7	<p>“I show either of two ‘graphs’ during the run. These are done using the standard ASCII block graphic characters in four colours to give about 16 levels of colour/shading, from full red for high values to full blue for the lowest.</p> <p>One shows the total number of positive bits in each gene over the whole population. This show me which genes have fully converged and which bits still have high variation.</p> <p>The other ‘graph’ shows the actual gene values (ranged from minimum to maximum) to show which values are being favored and which values are dropping out.</p> <p>These graphs allow me to glance at the screen and decide whether it is worth stopping, or whether it should run a bit longer. The latter also indicates whether some values are still at a high level even if not present in the top 5 shown individually.”</p>
A5	“best solution achieved every generation”

the family tree of a chromosome. One of the members of the research group also suggested showing an individual chromosome's family tree, another suggested showing the occurrence of niches within the population (i.e. clusters of similar chromosomes). The responses from the applications group in addition to viewing information regarding the algorithm and its population data, also suggested the visualization of problem-specific data such as the chromosome's phenotypic features.

B.4 Interaction Opportunities

Question 10

Having gained an insight into what could be viewed, Question 10 went on to ask the users how helpful, or destructive, they would find some example interface mechanisms for interacting with their GAs. These included the use of a bi-directional video-style control panel (see Tables B.50, B.51 and B.52), an algorithm parameter editor (see Tables B.53, B.54 and B.55), a chromosome editor for editing the chromosomes in the current population (see Tables B.56, B.57 and B.58), and for editing chromosomes in the reproduction gene-pool (see Tables B.59, B.60 and B.61).

Table B.50: Question 10.1. The reported opinions of the members of the theory group on the use of a bi-directional execution control panel.

RESPONDENT	COMMENTS
<i>T2</i>	"Can already do it."
<i>T3</i>	"Very useful."
<i>T1</i>	"Best off just grabbing all the data and looking at it later"

The use of a bi-directional control panel was considered extremely useful (Tables B.50, B.51 and B.52). Fifteen of the nineteen respondents noted advantages for a bi-directional control panel (group divisions; theory 1/3, research 7/8, applications 7/8) and 0 noted any disadvantages.

Parameter editing during execution was also generally seen as a useful form of interaction, 15 of the respondents noted some advantages, only 1 noted any disadvantages (see Tables B.53, B.54 and B.55, group divisions advantages; theory 1/3, research 8/8, applications 6/8, disadvantages; theory 0/3, research 0/8, applications 1/8).

Table B.51: Question 10.1. The reported opinions of the members of the research group on the use of a bi-directional execution control panel.

RESPONDENT	COMMENTS
<i>R6</i>	“For my use - very limited.”
<i>R8</i>	“Useful for debugging, but not in production runs.”
<i>R7</i>	“Again, fun, and good for teaching but that is all.”
<i>R1</i>	“very useful - like an omniscient, but impotent viewer.”
<i>R5</i>	“Excellent.”
<i>R2</i>	“extremely useful”
<i>R4</i>	“All of those options (bar one) are catered for in GAMeter, so I think they are useful! :)”
<i>R3</i>	“This is/was quite useful for me. During the solution of my first problems I needed such a control panel and thus implemented one in Matlab. If the computing power is high enough or the problem simple, this online control is useful. However, now most of my problems take hours of computing time. Thus, I run the GA offline and save all (intermediary) results.”

Table B.52: Question 10.1. The reported opinions of the members of the applications group on the use of a bi-directional execution control panel.

RESPONDENT	COMMENTS
<i>A5</i>	“would be very good”
<i>A8</i>	“useful”
<i>A2</i>	“HELPFUL”
<i>A6</i>	“Very helpful (a definite)”
<i>A3</i>	“This would be extremely useful”
<i>A1</i>	“We have already implemented “start” and “stop” and interactive changing of the parameters.”
<i>A7</i>	“I find it valuable to be able to stop at the end of any generation, then I can save the current complete set when paused and often save intermediate stages in important runs. I have never felt any need to step back to a previous generation. I have an option to store the best individual from every generation in a file, so that I can view the whole run and see which genes stabilized early, and which settled down later in the run.”

Table B.53: Question 10.2. The reported opinions of the members of the theory group on the use of an editor to change their algorithm's parameters during execution.

RESPONDENT	COMMENTS
<i>T2</i>	“Can already do it”
<i>T3</i>	“Very useful.”
<i>T1</i>	“no”

Table B.54: Question 10.2. The reported opinions of the members of the research group on the use of an editor to change their algorithm's parameters during execution.

RESPONDENT	COMMENTS
<i>R3</i>	"see above [10.1]. Most of the time you don't have to. Nevertheless, some problems are easier to solve, when you change parameters during optimization. For this, you have to understand, what's going on. With my control panel I could change the parameters on the fly, even without breaking/stopping the calculation - should be useful in a control panel"
<i>R7</i> <i>R1</i>	"Again, fun, and good for teaching. However, probably disruptive to GA. Hard to say." "Could cause problems, but I think it would be really interesting, as the user could get the chromosomes away from local maxima, which is exactly the sort of thing humans are good at. Also simulates a kind of real environment, which changes over time, and could test chroms ability to adapt in a changing environment, maybe."
<i>R8</i> <i>R6</i> <i>R4</i> <i>R5</i> <i>R2</i>	"Maybe useful if the GA gets stuck." "Useful - if the GA is not converging, altering the mutation rate could help." "Again this is useful. Often the GA can be improved if the parameters are adjusted during run-time. (spoken form experience!)" "Definitely a good idea." "e. useful"

Table B.55: Question 10.2. The reported opinions of the members of the applications group on the use of an editor to change their algorithm's parameters during execution.

RESPONDENT	COMMENTS
A5	"not very good idea, instead an adaption scheme can be developed."
A8	"of some interest"
A3	"This probably depends on the type of problems to solve, it would probably help for large problem that only need to be solved once"
A2	"HELPFUL"
A6	"Could be quite useful for me as I will also be using a GA to identify the mechanical system. If I obtain a 'better' parameter set for my dynamic system whilst the main GA is running it would be useful to be able to introduce this to the current run."
A1	"This can be very useful, to speed up the algorithm and to increase the user's insight into the process."
A7	"I can do this at any time when paused, (and I an only pause between generations), although I rarely do so, except sometimes to lower the settings during a run as there is no point (in my case) in running a GA in a changing environment."

Table B.56: Question 10.3. The reported opinions of the members of the theory group on the use of an editor to alter the population's chromosomes between two generations.

RESPONDENT	COMMENTS
T1	"no"
T2	"Could do it, but can't see any reason to do it."
T3	"An intriguing but strange idea; like getting Fred next door to do brain surgery on you by trial and error with a soldering iron."

Table B.57: Question 10.3. The reported opinions of the members of the research group on the use of an editor to alter the population's chromosomes between two generations.

RESPONDENT	COMMENTS
<i>R8</i>	"See earlier remarks." [10.2]
<i>R1</i>	"Why? this seems silly and only useful for initial testing of the program. I must be missing the point."
<i>R3</i>	"Huhh, what are mutation and recombination and so on for? If the operators are good, you don't should do this. If not, change your operators or look for a better mapping/embedding of the problem."
<i>R6</i>	"For my use - limited."
<i>R5</i>	"A creative idea, yes I'd like to try that (though I hesitated a moment). Yes, real biologists as well as observing, they do experiments like stealing a lion's cubs to observe the reaction. We should certainly be able to do that."
<i>R7</i>	"Again, fun, and good for teaching. I suppose it could be used "heuristically" to change direction of search, but this is an ad-hoc approach to an already stochastic optimization technique."
<i>R2</i>	"e.e. useful"
<i>R4</i>	"Useful (and yes, in GAmeter!)"

Table B.58: Question 10.3. The reported opinions of the members of the applications group on the use of an editor to alter the population's chromosomes between two generations.

RESPONDENT	COMMENTS
<i>A3</i>	"no"
<i>A1</i>	"I don't really see the use of this. But perhaps I'm missing something here. It wouldn't hurt as an option, that's for sure"
<i>A5</i>	"not good idea, that would interfere in GA's search strategy."
<i>A8</i>	"of minor value"
<i>A2</i>	"HELPFUL"
<i>A6</i>	"it may be useful to be able to input new chromosomes during the run to allow for expert knowledge to be incorporated"
<i>A7</i>	<p>"I find it very useful to be able to edit the chromosome. This is often done to compare my intuition with the current settings, or to to check whether small variations in the current optimum would further improve it. More usually I find out why my intuition would give a worse answer. In some cases if my graphs indicate that certain values are not being used I can seed the population with an individual with these values and see if it can spread these genes in future generations.</p> <p>I do the editing by using the current best chromosome as a default, then the edited chromosome replaces the current worst individual in the population."</p>

Editing the populations' chromosomes between generations was also considered useful but also counter-intuitive to the intention of autonomous evolution, (see Tables B.56, B.57 and B.58). Overall 11 of the 19 respondents noted some advantages and 4 noted some disadvantages. The distribution between the groupings was as follows, advantages; theory 0/3, research 7/8, applications 4/8, disadvantages; theory 1/3, research 2/8, applications 1/8.

Table B.59: Question 10.4. The reported opinions of the members of the theory group on the use of an editor to alter the gene-pool's chromosomes within a generation.

RESPONDENT	COMMENTS
<i>T1</i>	"no"
<i>T2</i>	"Can not see any good reason to do it."
<i>T3</i>	"Pedagogically nice, I suppose."

The idea of editing the gene-pool's chromosomes within a generation was noted as a strange idea and several of the respondents couldn't see any advantage to such an enterprise, although some noted that it may be pedagogically useful (see Tables B.59, B.60 and B.61). Seven of the respondents noted some advantages (group divisions; theory 1/3, research 3/8, applications 3/8), and 2 noted some disadvantages (theory 0/3, research 1/8, applications 1/8).

Question 11

Here the users were asked to add any other form of interaction that they would consider beneficial, their comments are included in Tables B.62, B.63 and B.64.

B.5 Any Other Comments

Question 12

Finally the users were asked to add any other suggestions as to how GAs could be made easier to use, their comments are included in Table B.65.

Table B.60: Question 10.4. The reported opinions of the members of the research group on the use of an editor to alter the gene-pool's chromosomes within a generation.

RESPONDENT	COMMENTS
<i>R5</i>	“Personally I cannot see myself bothering with that. I can't see that helping to study <code>_macroscopic_</code> population behavior, which is the important thing.”
<i>R1</i> <i>R6</i>	“Ditto” [see 10.3] “I use steady-state GAs, so the gene-pool's chromosomes within a generation. I use steady-state GAs, so the gene-pool is the same as the population.”
<i>R3</i>	“You divide between population and reproduction gene-pool. I am not sure, that I get the difference. My populations are my reproduction gene-pools. Am I missing something?”
<i>R4</i>	“Hmmm, I'm not so sure of this one. Since there is an operator which decides which solutions enter the population pool, so you need to edit the reproductive pool? (Especially if you can edit the population pool)”
<i>R8</i>	“See earlier remarks.”
<i>R7</i>	“Again, fun, and good for teaching. However, probably disruptive to the GA. Hard to say.”
<i>R2</i>	“e.e. useful ++”

Table B.61: Question 10.4. The reported opinions of the members of the applications group on the use of an editor to alter the gene-pool's chromosomes within a generation.

RESPONDENT	COMMENTS
<i>A3</i>	"no"
<i>A1</i>	"Same as 10.3, don't see the use at the moment."
<i>A8</i>	"of minor value"
<i>A7</i>	"I only edit when paused at the end of a generation. I can't think of any reason to stop during a generation."
<i>A5</i>	"not a good idea"
<i>A6</i>	"see above." [10.3]
<i>A2</i>	"HELPFUL"

Table B.62: Question 11. The other forms of additional interaction reported by the members of the theory group.

RESPONDENT	COMMENTS
<i>T3</i>	"The ability to reinitialise the population in any of various ways at one's chosen time. Altering things like penalties for the cost function."

Table B.63: Question 11. The other forms of additional interaction reported by the members of the research group.

RESPONDENT	COMMENTS
<i>R8</i>	“Ability to introduce a strong mutation pulse to kick the GA into a different region of solution space.”
<i>R2</i>	“an AI module which monitors the ‘directors’ behavior and learns how to direct the GA itself (only joking (well 75% joking!!))” (AI = Artificial Intelligence)
<i>R1</i>	“changing population size, and even chromosome size might be useful.”
<i>R4</i>	“Well, there’s problem specific interaction. For example changing a problems variables - or displaying the solution graphically which you can only do with some problem knowledge there. - There’s displaying (not really interacting) a series of results from a set of experiments. Useful in seeing how(if) consistent the GA is. I guess the list is endless, but there is a limit on how useful all these interactions are.”

B.6 Future Contact

Question 13

The last section of the questionnaire asked the users if they would object to being contacted in the future with reference to this project and the evaluation of the resulting GA visualization system.

Only 2 of the 19 respondents objected to being contacted in the future.

Table B.64: Question 11. The other forms of additional interaction reported by the members of the applications group.

RESPONDENT	COMMENTS
A7	<p>“In order to evaluate why a particular individual is the best (compared with my own ideas) I have a full 50 line screen of data showing intermediate calculations used in the evaluation function. This is essential to determine the effect of minor (and major) changes to the current settings, to show the effect of each gene in the whole picture, and to allow me to explain why the best individual is better than other alternatives. This type of display is obviously specific to any particular problem. However, any program with an evaluation function should be able to show specified intermediate calculations in that function.</p> <p>While in the edit mode I have the option to look at any single gene, or any pair of genes to see what values occur with changes over the full range of these genes (with all other genes held constant). This helps to check how much influence a given gene has on the current system, as well as checking whether it is at the true optimum. The 2-gene system is particularly useful here, but I can't think of a good way of showing 3 or more genes at once.</p> <p>The above display can either show the actual values, or use a 16 shade graph as described previously. The 2-gene graph often shows diagonal ridges, where changing any single gene gives a worse rather than better result, whereas changing both genes can lead up the diagonal ridge to better values. I presume the same diagonal ridges occur in higher dimensions.”</p>
A5	<p>“initial partial seeding of population with some “good” chromosomes (using domain knowledge).”</p>

Table B.65: Question 12. All of the received additional suggestions for making GAs easier to use.

RESPONDENT	COMMENTS
Theory Group <i>T3</i>	<p>“On GAs in general, I have too many comments to give and not enough time. 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.”</p>
Research Group <i>R8</i>	<p>“The really tricky issue is designing the representation and the operators, not controlling or visualizing the GA during running.”</p>
<i>R3</i>	<p>“The implementation of a visualization tool used by many people is quite difficult. If you could define a really portable format for the data... I would like to hear more about your thoughts.”</p>
Applications Group <i>T1</i>	<p>“I’d like to see any tool be applicable more broadly than just to GAs, but then I don’t care too much about chromosomes and all that.”</p>
<i>T2</i>	<p>“Any such GA package needs to be able to show visualisation of individual runs, and gather statistical info on batches of runs. Also the representation of genes and selection of which reproduction operators must be easily changed.”</p>

Appendix C

GA User Questionnaire Responses

This appendix presents the individual anonymised responses received in the GA user study presented in Chapter 3.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - *T1*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

T1 : 7 years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

T1 : Research in: solving TSP, distributed GA's and the effect of the underlying topology.
Comparative studies of representation spaces.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

T1 : Interest in evolution. Interest in seeing how well GAs work on such problems. General research in optimisation and landscape structures.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

T1 :

Unix

gcc 2.6.3

C++ language

Tcl/Tk for scripting and interfaces

Tcl-dp for distributed purposes

Plplot for graph drawing (not much yet)

also macintosh with symantec c++ for development purposes

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

T1 : This is to my mind the most important step in any algorithm, perhaps more important than the choice of algorithm.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

T1 : I'm less interested in this – as I generally look at pretty precise TSP problems or whatever, and investigate landscapes and other such things genetic

*Q.5.3: CHOOSING THE GA's COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

T1 : genetic operator very important and hard to pick. The rest ain't too important in my opinion

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

T1 : No opinion

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

T1 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

T1 : Extract loads of data during runs – not very efficient, but I want to know as much as possible.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

T1 : Compare with other algorithms, known bounds etc. This is important to me.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

T1 : none for my purposes

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

T1 : too much to look at

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

T1 :

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

T1 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

T1 : Need more than this; need to know the local structure of fitness changes throughout the population.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

T1 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

T1 : I'm more interested in more general questions and different algorithms, so this ain't much use to me.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

T1 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

T1 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

T1 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

T1 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

T1 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

T1 :

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT

POSITIONS (“LOCT”) MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

T1 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

T1 : The ratio of accepted changes to non-accepted, the correlation in fitness values as a function of distance in representation space, the frequency of local fitness peaks,... comparisons with other algorithms.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAS?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

T1 : Best off just grabbing all the data and looking at it later

*Q.10.2: EDITING THE ALGORITHM’S PARAMETERS DURING EXECUTION:

T1 : No

*Q.10.3: EDITING THE POPULATION’S CHROMOSOMES BETWEEN TWO GENERATIONS:

T1 : No

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL’S CHROMOSOMES WITHIN A GENERATION:

T1 : No

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

T1 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

T1 : I'd like to see any tool be aplicable more broadly than just to GAs, but then I don't care too much abotu chromosomes and all that.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

T1 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - *T2*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

T2 : 3 years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

T2 : Research on representation and role of mutation. Self adaption and solving specific problems.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

T2 : Research

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

T2 : Smalltalk V on Dos.

Gnu C++ on unix

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

T2 : String representation is limiting. Not useful for all problems

Better representations exist.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

T2 : Only when there are conflicting criteria.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

T2 :

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

T2 : This is a difficult problem, as parameter settings drastically affect the efficiency of the GA.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

T2 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

T2 : Pick an instance of the problem with a known solution, so that you can verify that it can be found. Then gather statistics on solving the problems over a number of runs.

IS THERE ANY OTHER WAY?

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

T2 :

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

T2 : Can do it already

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

T2 :

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

T2 : See above

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

T2 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

T2 : Can do it already. Shows you what is happening during single runs.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

T2 : Slow down the run

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED

FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

T2 : Can do it. Only useful to check if GA works correctly, and see the effect of hamming cliffs

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

T2 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

T2 : The showing the occurrence of mutation by itself is not very useful. Percentage of mutations that are better than its "parent" are useful to show the effectiveness of mutation over the run.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

T2 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

T2 : Can't see any, so have not done it.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

T2 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

T2 : Only bother in terms of fitness. This would be seen by the average fitness being very close to the maximum fitness.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

T2 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

T2 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

T2 : Can already do it.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

T2 : Can already do it

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

T2 : Could do it, but can't see any reason to do it.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

T2 : Can not see any good reason to do it.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

T2 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

T2 : Any such GA package needs to be able to show visualisation of individual runs, and gather statistical info on batches of runs. Also the representation of genes and selection of which reproduction operators must be easily changed.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

T2 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *T3*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

T3 : About 4 years.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

T3 : Various timetabling and scheduling problems, real and contrived. Facility layout problems, set-covering problems, pipe-routing, and various miscellany.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

T3 : Primarily because evolutionary algorithms are my principal research interest. For practical problems, they promise flexibility and fast prototyping, though not necessarily best results of course; this very point is part of my research, however.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

T3 : UNIX, C

DOS, C

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

T3 : As a researcher, I don't find this difficult, so much as I find it a fascinating arena for

experimentation. Nevertheless, in many senses this is perhaps the most difficult bit, since I tend to work in areas where there is endless possibility for this mapping.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

T3 : I don't tend to find this as crucial as 5.1, but then again that's probably because the design of this usually follows fairly directly from it.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

T3 :

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

T3 : My considered view is that, interesting as the space of possibilities are here, time spent on other matters tends to be far more fruitful than endless tuning of the parameters and components. So I tend to use fixed overall choices for these, subject to change at whim, or following recent results found in the literature. Operators are really a different matter from the other things in these two questions though; it's within these that you can stick appropriate domain specific knowledge, or hybridise with other ways to solve the problem.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

T3 : Defining one or more distance metrics between genotypes and/or phenotypes is often appropriate for various reasons. Also, my GAs tend to be parametrisable to SA too, so components are required which enable me to view acceptance rates at different temperatures, so as to establish a good initial temperature.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

T3 : Usually there are benchmark results available; if not, then always compare with SA and various kinds of hillclimbing.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

T3 : Not sure I understand this, although I tend to favour the production of multiple distinct solutions. This is mainly where the distance metrics come in. The results are better to the extent that there are multiple solutions with a good average distance between them.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

T3 : You can see what's going on ! In some cases of course – like photofit generation, or evolving art, this is necessary anyway.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

T3 : You can see what's going on ! In some cases of course – like photofit generation, or evolving art, this is necessary anyway.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

T3 : Very flexible, to the extent that the user requirements can be varied.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

T3 : Could be easy to hide what's really happening.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

T3 : People tend to like this simply because it shows there's something actually happening. If a 'when fitness get's here we're fine' line is on the graph, possible for most problems, then the illusion of understanding the GA's progress is comfortably strong.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

T3 : None really, except there might be much messing around on some problems to translate fitnesses into graphable quantities – eg: if half the pop is between 0.00001 and 0.00002, with others around 21,345,789,329.6

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

T3 : In one sense, this doesn't tell you much more than the and visualisation of the entire pool (or bits of it), provided you already know what the selection pressure roughly is. It is occasionally interesting when a very lowly fit chromo is chosen for parenting, but you already know that will happen every now and then. What's interesting, though, is when poor parents lead to strong children. I had some success in a program which indicated on screen every example of a crossover

yielding a child better than both parents. Among other things, this is good to help users justify the use of a GA to their bosses.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

T3 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

T3 : Entertaining

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

T3 : You need to be very selective, since there are so many.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

T3 : Entertaining

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

T3 : Surely gets very dull after a while, so maybe use only as an option for demos, new users, etc.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCI") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

T3 : As one of potentially many such diversity measures, this is certainly an important thing to

show. Helps much in seeing what's going on

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

T3 : Needs more sophistication to be truly useful. Eg, there may be several best-fit chromos, all genotypically or phenotypically distinct.

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

T3 : Generally, info on a variety of interesting events – eg: every time a new best fit arrives, let's see its parent(s) and the operation which produced it. Let's also see those operations in which very good parent(s) led to terrible children. Good also to see what's occurring in and between niches. Eg: an ongoing measure of how child fitness correlates with parent diversity.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

T3 : Very useful.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

T3 : Very useful.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

T3 : An intriguing but strange idea; like getting Fred next door to do brain surgery on you by trial

and error with a soldering iron.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

T3 : Pedagogically nice, I suppose.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

T3 : The ability to reinitialise the population in any of various ways at one's chosen time. Altering things like penalties for the cost function.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

T3 : On GAs in general, I have too many comments to give and not enough time. 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.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

T3 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - *R1*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R1 : about 2 months

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R1 : playing Prisoner's Dilemma

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R1 : I am a computer science student at Cambridge University, and for my project I am working with relating GAs to visual images of creatures, kind of like Todd and Latham's stuff.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R1 : 486 SX 25 MHz : MsDOS, Turbo C++

Solaris V UNIX gcc compiler

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R1 : in the case of Prisoner's Dilemma this is really easy. I imagine that the mapping onto geometric objects will be much harder

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R1 : Again, for PD this is quite easy. The problem I am having is that I want to set up a test for diversity, and I don't know much about statistics so I have had to make it up as I go along, and I'm not convinced that it's very good.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R1 : Initial population creation is purely random, so no problem there. Selecting parents is where I have a lot of problems. At the moment I have it so that about the best 10% produce offspring, mating with randomly chosen partners, but I am finding it a pain to come up with a way to make the number of offspring proportional to score - particularly since I have a fixed population size.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R1 : This is the biggest problem for me. Population size I have figured out by experimentation needs to be around 100, otherwise nothing good ever develops. Finding a good mutation rate is a nightmare. At the moment I have a mutation rate changing on the fly, to try to help wipe out large populations of the same thing.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R1 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R1 : The chromosomes play 10 games, each game being 100 rounds against randomly selected opponents from the population. the points add up to give a fitness measure. This is then multiplied by a measure of diversity give an overall score.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R1 : using the diversity test - here I compare the values of the genes in each chromosome with the average value for each gene and the higher the difference, the greater the score.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R1 : Massive amount of information!

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R1 : Too much information for a human to usefully digest.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R1 : Limit the number of things the user has to examine.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R1 : The user may not know what he/she is doing, and could pick anon-representative selection, and miss the interesting things.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R1 : See how quickly the chromosomes converge on a solution, also see how stable populations are.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R1 : only useful in conjunction with the other graphs showing fitness, otherwise for example, you could get graphs which bottom out after n generations, but don't tell you how well the populations were actually scoring, just the fact that they had reached a stable state.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R1 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R1 : Way too confusing - I don't see how a user could get any useful information out of such a picture. Perhaps I'm wrong. Actually, I guess that in the testing stages, when the population size could be kept small, it would be reasonable.

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R1 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION

OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R1 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R1 : Would be interesting, I guess!

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R1 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R1 : Aha! This is very good. I think it is necessary in populations that tend to stabilize, in order to keep variation going (if that's what you want) but if you just want to solve a problem that could also be useful to push away from local maxima.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R1 : As I said above, I don't really know how to implement this sort of thing - it sounds like statistics to me. Could be quite computationally expensive, maybe?

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT

YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R1 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R1 : very useful - like an omniscient, but impotent viewer.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R1 : Could cause problems, but I think it would be really interesting, as the user could get the chromosomes away from local maxima, which is exactly the sort of thing humans are good at. Also simulates a kind of real environment, which changes over time, and could test chroms ability to adapt in a changing environment, maybe.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R1 : Why? this seems silly and only useful for initial testing of the program. I must be missing the point.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R1 : Ditto.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R1 : chaning population size, and even chromosome size might be useful.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R1 : I hope that my project will end up with a way to visually display chromosomes (ie the phenotype) and see how fitness of the genes to solve on sort of problem relates to their appearance. Perhaps I'll let youknow if I get any interesting results!

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R1 : no

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R2*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R2 : 1 year

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R2 : Research, mainly in neural net construction

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R2 : Seemed like an interesting idea at the time ;-)

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R2 : C++, using custom-written graphics to display network weight values, and errors, etc.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R2 : Only a problem when there are lots of constraints

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R2 : Difficult when the final fitness is a function of a number of attributes. E.g., if you want to minimise cost, while maximizing productivity, while...

Also, when the fitness landscape is very flat, apart from a few localised peaks. A while ago I tried to get a GA to come-up with a XOR circuit, using OR, AND, and NOT (I think?). Anyway, solutions which were very close to a solution were really unfit. I gave up trying to design an effective evaluation function.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R2 : Creating initial population can sometimes be time-consuming when there are a number of constraints. I would then try to design an evaluation function which would not violate these constraints, given two 'legal' strings. The XOR problem above is a good example.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R2 : Creating initial population can sometimes be time-consuming when there are a number of constraints. I would then try to design an evaluation function which would not violate these constraints, given two 'legal' strings. The XOR problem above is a good example..

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R2 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R2 : Just by assessing the fitness functions

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R2 :

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R2 : If clusters were forming around local minima

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R2 : Depends on the population size I suppose. Execution speed might be a problem, e.g. for a 'director' who wished to observe how the population changed overtime. Slow updates might make it more difficult (from a cognitive) perspective to observe this.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R2 : as its user-defined I can't foresee any problems PROVIDED the user know what subset of strings he/she wants, and how to specify them

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R2 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R2 : Good to see if population is stagnating, and might need a boost (e.g. load of mutation, or a few new randomw strings)

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R2 : can't think of any

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R2 : probably not a) the easiest thing to do in general b) the most useful but probably is problem-dependent to a large degree

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R2 : could get tricky!

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R2 : dunno

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R2 : dunno

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED

TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R2 : Probably good for education purposes, but when you think of the number of matings that are going to occur in a typical GA run, then it'd probably be too much to take in.

HOWEVER, if your mating function were in some way 'intelligent', then you, the director, may want to observe how the crossover (or selection process) was being decided/performed.

For example, you might have a selection process which (instead of the traditional, biased roulette wheel) involved the strings wandering around a grid until they find a mate they fancy and then reproduce. The 'attraction' function might be an evolving AI module of somekind, and you might want to observe it working.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R2 : see above

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R2 : good to show clustering around peaks

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R2 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R2 : It might be interesting if you could look at a 'family tree' of an individual chromosome, and see how the fitness improves.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R2 : extremely useful

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R2 : e. useful

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R2 : e.e. useful

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R2 : e.e.useful++

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R2 : an AI module which monitors the 'directors' behaviour and learns how to direct the GA itself (only joking (well 75% joking!!))

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R2 : Saving a GA run, and replaying it at a later date? Sorry, I've exhausted myself Trevor... Hope this is of interest.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R2 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R3*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R3 : Nearly 2 years.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R3 :

Optimization of systems

- standard test suite for GA's
- real world problems (greenhouse control, satellite movement)
- dynamic optimization problems

My first task was the development of a Genetic Algorithm Toolbox for Matlab (during my time in Sheffield one year ago). This toolbox is available from me.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R3 : The implementation of GA's is straightforward. They are powerful. Using, for instance, gradient based methods, is often not possible for real world problems.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R3 : Matlab - on different computer systems (PC and SUN Sparc). If you don't know Matlab: this is a powerful programming and visualization environment available on nearly every computing platform. The development time for a system is short, because of the huge number of problem

specific toolboxes. Especially in control Matlab is widely used. (The drawback: Matlab is expensive. However, most university own site licenses - have a look to the control group).

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R3 : No real problem. The Toolbox can use real and binary variables. Until now, all of my problems used real parameters. However, I know, that there are a lot of problems, were the mapping/embedding is difficult.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R3 : Here goes the work. 80%-90% of the time for programming/solving the problem is needed for implementing the evaluation function.

*Q.5.3: CHOOSING THE GA's COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R3 : If I don't define special parameters the Toolbox uses default parameter . This includes every part of the algorithm. Thus, if I don't know a lot about the system, I work with the default ones. On the other side, I can change everything. However, most of the time I don't have to.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R3 : see above

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS TAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU

ENCOUNTER BELOW.

R3 : Not at the moment. However, I think about implementing something, but I didn't find a clean and general way of doing it. Every system is different. On the other side, quite a few problem need a sophisticated preprocessing. This could speed up the optimization considerably.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R3 : Have a look to the data/results. You have to understand the problem, otherwise you can't weight the solution of the GA. Or, when you get results try to understand them - this is often the way to learn more about your system. I didn't find a global way for weighting. The given best solution of the GA depends very much on the evaluation function.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R3 : see above. If the evaluation function is smooth, you don't need such a long run of the GA, if it is more chaotic - then you have a problem and you are lost in the GA-space.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R3 : if only one generation (the actual one) at once

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R3 : too much information, you don't have to see every bit of information.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R3 : necessary, if you know/understand the overall meaning of your data, you want to have a look. For instance, I plot the chromosome of the best individual in every generation over all generations. Thus, I get a meaning of the change of the best individual during the optimization.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R3 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R3 : If you plot the fitness of the population (only best individual and/or mean and/or worst), this plot includes the gradient. Thus, normally it is not really necessary if you have the fitness directly. However, one of them is absolutely necessary.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R3 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R3 : see 8.3

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R3 : see 8.3

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R3 : see 8.3

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R3 : see 8.3

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R3 : If I want to know, how the GA works, this would be useful. but see below.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R3 : This would be much to operator specific. Using GA I want to solve problems. I don't wanna know how the GA works.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R3 : Similarity to the fittest sounds like a good idea. (I do an histogramm of the differences between all individuals between each other. This gives a meaning of the diversity of the population.)

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R3:

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R3:

At the moment I visualize the following things (every 10 generations or so):

- fitness value of best individual in the last 20-40 generation
- chromosome of best individual in the last 30-60 generation
- all chromosomes in the actual generation
- all fitness values in the actual generation
- histogram of diversity of chromosomes in actual generation (first try, needs more work)

This is quite enough for a good understanding, what's going on.

(For every system I often include system specific visualizations (dynamic optimization -; results of simulation with best individual for instance.)

There are lots of new possibilities, if you can make movies and so on. At the moment, the computing power is far too less to think about an implementation. However, I think there should a lot be done. I will do some thinking as well and when you contact me, we can talk about more ideas.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R3 : This is/was quite useful for me. During the solution of my first problems I needed such a control panel and thus implemented one in Matlab. If the computing power is high enough or the problem simple, this online control is useful. However, now most of my problems take hours of computing time. Thus, I run the GA offline and save all (intermediary) results.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R3 : see above. Most of the time you don't have to. Nevertheless, some problems are easier to solve, when you change parameters during optimization. For this, you have to understand, what's going on. With my control panel I could change the parameters on the fly, even without breaking/stopping the calculation - should be useful in a control panel.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R3 : Huhh, what are mutation and recombination and so on for? If the operators are good, you don't should do this. If not, change your operators or look for a better mapping/embedding of the problem.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R3 : You divide between population and reproduction gene-pool. I am not sure, that I get the difference. My populations are my reproduction gene-pools. Am I missing something?

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R3 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE

EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R3 : The implementation of a visualization tool used by many people is quite difficult. If you could define a really portable format for the data...

I would like to hear more about your thoughts.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R3 : I would appreciate being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R4*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R4 : approx. 2 years.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R4 : Various optimisation problems, particularly very difficult problems in the real world!.

Designing a GA toolkit, GAMeter. I have been using GAs because the problem domain can be separated from the search domain, hence generic toolkits are possible.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R4 : Traditional techniques, if they exist, are extremely computationally expensive to use on the size of problems that I am using. GAs (and other heuristics, such as SA, TS, etc.) seem ideal for these class of problems.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R4 : UNIX machines (various) / PCs

Language: C

Toolkit: GAMeter - a generic GA toolkit developed at UEA. It has a user-interface with many of the facilities you mention below. Problems can be integrated into GAMeter VERY easily and every parameter can be changed interactively even at run-time.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP

STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R4 : This is usually the most important stage and will more than usual, dictate the ease (or lack of) that the following steps will be implemented. This is, arguably, where the clever thinking is required when using a GA - that or luck.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R4 : This will wither drop out from the problem objective or the representation. If this is not the case, then this stage may be harder than necessary requiring some subtle technique to return a fitness value.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R4 : Again, most of these components will be apparent from the representation. For a binary bit string, I use basic operators and see how they perform. For permutation problems, then operators get tricky.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R4 : I use very basic parameters initially and see if the GA will work using a 'dumb' GA. As I have said, with GAmeter, it is very easy to change parameters at any time. Thus the initial parameter settings does not worry me too much as I know they can be changed at will.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS TAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R4 : For most of the problems I have tried, No. As mentioned I use a dumb GA. Try it out and

see what results I get. If it is not working, then a re-think of the representation may be required. If it works OKish, then I play around with parameters, possibly new operators, to see if there is any improvement, and how much.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R4 : Depending on the problem in question. Several alternatives exist...

- Try against an exact method if one exists for quality of solution.
- Try against a specialised heuristic.
- Try against a general heuristic, SA, TS, etc.

The list is endless really. As I said, it will depend on the problem in hand.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R4 : This is an academic question really. I apply GAs in an industrial context, where the quality of the output is more important than how it fares to all other points in the search space. Hence I usually concentrate on the above step.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R4 :

- This is good for seeing how similar (or not) all members of the population pool are.
- It may suggest ways in which improvements could be made to the GA (for example niching)
- It may highlight how the GA has become trapped in a local optimum.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R4 :

- It can be confusing if you have very large bitstrings. (I sometimes work with bitstrings of '000s bits) - not very informative if the genotype/phenotype map is not straightforward.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R4 :

- reduces disadvantage #1. (If I understood you correctly!)

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R4 :

- How do we know the selection is a fair selection?
- Gives another burden to the user to decide.
- doesn't help disadvantage #2.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R4 :

- Standard visualisation tool everyone can understand.
- shows convergence of GA, etc.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R4 :

- Too much emphasis can be placed on the graph without going into any detail as to why that pattern occurred.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R4 :

- Shows how the genetic operators have been working
- Shows which children were automatically discarded.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R4 :

- Is this needed?

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R4 :

- Can highlight whether mutation should be increased or decreased.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R4 :

- A lot of overhead for this information.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R4 :

- can highlight when the operator has become defunct or when another operator would be more useful.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R4 : as above.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R4 :

- could be usefull...

*Q.8.4.D: A “SIMILARITY” RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS (“LOCF”) MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R4 :

- but it really depends on the problem. For example, in highly epistatic problems, its not the number of different genes but the actual genes that are different which determines how good or bad that solution is. This information would be useless in this case.

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R4 : This I am very interested in. As a developer of a toolkit, I am always looking at ways in which the visualisation could be improved. But at the same time I think about the overhead caused by this visualisation.

My outlook is visualisation is nice, but not for the sake of speed and general usefulness. (i.e. it's no point added some functionality if most problems don't need this information or visual guidance.)

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R4 : 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 thats a lot of overhead on the GA.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R4 : Again this is very useful. Often the GA can be improved if the parameters are adjusted during run-time. (spoken from experience!)

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R4 : Useful (and yes, in GAmeter!)

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R4 : Hmmmm, I'm not so sure of this one. Since there is an operator which decides which solutions enter the population pool, so you need to edit the reproductive pool? (Especially if you can edit the population pool)

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R4 :

- Well, there's problem specific interaction. For example changing a problems variables - or displaying the solution graphically which you can only do with some problem knowledge there.

- There's displaying (not really interacting) a series of results from a set of experiments. Useful in seeing how(if) consistent the GA is.

I guess the list is endless, but there is a limit on how useful all these interactions are.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R4 : If you are interested in seeing GAmeter, you are more than welcome to. It is free for academic purposes.

Perhaps it may give you a few more ideas, or more likely, you can suggest future improvements.

GAmeter is continuously evolving and I am always on the lookout for new ideas, etc.

It sounds as if you have already thought about many of the options that are already included.

email me if you are interested... jwm@sys.uea.ac.uk

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R4 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - *R5*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R5 : Six months (studying them 2 years).

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R5 : Standard cell placement. It's a small part of the problem of designing silicon chips.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R5 : My interest in GAs comes first. Somebody suggested the placement problem as a hard optimization problem that might even have some money in it. As to why I like GAs... that's really because as you study the subject, your mind is throwing up ideas for improvements almost as fast as you understand it. That is, the subject is young and scruffy.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R5 : HP 700 series work stations...

Model 710 712/60 720 735/125

Ram 48Mb 64Mb 64Mb 144Mb

Disk 500Mb 1Gb 500Mb 1Gb

735 is 2-3 times faster than the others.

I'm writing in Ansi C, I'm hoping to move over to C++ "When I've time" to learn it properly. I'm

already trying to use the object oriented philosophy in C. For example, my large number of functions are partitioned into files, one file for each "class". I try to produce functions for as low a class as possible for the sake of reusability. For example, I have "solution" functions, but a solution is a huge ugly thing that is derived (by the measure) from an -ordering- (that is an array of integers 1-n in some order). I've made crossover specific to the ordering class, because that may become useful for other projects.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R5 : That's The Problem. That's what makes the use of GAs like the mensa test. Recognizing what the parental contribution should be, and then figuring out a representation that supports that. I've got a friend whose head of department hired someone to solve a problem using a GA. She seems to have used any-old representation, and now she, and her boss, go round telling people GAs don't work. (Don't worry, I've offered to help out if they send me more info.)

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R5 : For this project, yes, that took time. But that was "just programming". I've made at least one very silly mistake that wasted a lot of time. Actually, it was the sort of mistake that Object Oriented Design would have made impossible, which is why I'm mending my ways.

Also I think I could have made more use of existing packages in the electronics field. But so far I've had to work without input from electronics professionals, and my publication (pending) will state that I am addressing a simplification of the industrial problems. You'll be glad to know I'm about to start a PhD at York - in an electronics department.

*Q.5.3: CHOOSING THE GA's COMPONENTS, E.G. THE INITIAL POPULATION CREATION

METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R5 : Difficult isn't the word... my whole approach is based on a special kind of population seeding! i.e. population creation. As for the other GA components you mention, really I've stuck to somebody else's published details about an algorithm, which I am trying to improve upon.

*Q.5.4: SELECTING PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R5 : 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.

I don't know if it's possible, but I'd like to see your package built with the idea of users contributing add-on modules. However thoroughly you build the thing, when I use it I am going to want to add more, and I would want to send my modules to some centre of cooperation.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R5 : My approach is to use an assisting optimizer to produce a paradigm solution which is partially optimized. Then I produce a population from it by scrambling the solution with small changes to the gene values (the ordering problem is a high alphabet problem with a metric i.e. some of the alleles are "near" one another.) The aim is to concentrate search on a small part of the solution space which is yet expected to contain global optimum. So far the assisting optimizer is just another GA, but in other domains it might be a different optimizer.

Say the word and I will send you my paper. I presented it at the recent AISB conference on Evolutionary Computation.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R5 : I'm just comparing solution quality to that found by my rival's GA.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R5 : I'm running the GA repeatedly from different starting points (the random string optimized to produce the paradigm string). Then I'm comparing GA outputs in terms of fitness, absolute phenotype features, and phenotypic features considered more abstractly. I believe that my problem has a large number of global optima, that are the same in statistical profile but very different in how that profile is instantiated.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R5 :

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R5 : If you can understand it, your problem is too simple.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - AD-

VANTAGES.

R5 : Establishes conventions.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R5 : We do not yet know enough to establish such conventions, yet maybe your package should make a stand and be open to change. At the moment everybody does their own thing here.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R5 : Essential. Even I report it.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R5 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R5 : Could be used to spot convergence.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R5 : Your package should spot convergence for us. Get rid.

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION

OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R5 : None.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R5 : Distraction. User should know where mutation will occur.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R5 : Good for educational purposes. Also, -all- of your ideas for displays might turn out to be useful for debugging the GA.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R5 : None as long as its optional.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCI") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R5 : This justifies your enterprise. If I don't know this I don't know what my algorithm is doing (that premise currently true!)

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCI") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R5 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R5 : Niches. I mean sort strings so that similar ones are together.

By the way, I'd like string similarity to be pretty flexible, or at least particularly open to user add-ons.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAS?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R5 : Excellent.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R5 : Definitely a good idea.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R5 : A creative idea, yes I'd like to try that (though I hesitated a moment). Yes, real biologists as well as observing, they do experiments like stealing a lion's cubs to observe the reaction. We should certainly be able to do that.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R5 : Personally I cannot see myself bothering with that. I can't see that helping to study -macroscopic- population behaviour, which is the important thing.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R5 : Can't think of any others.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R5 : The above covers it I think.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R5 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R6*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R6 : 2 years.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R6 : Determining the best transmission scheme and data rate for a baseband communications system. Designing FIR filters. Producing bit sequences with special autocorrelation functions.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R6 : It was more a case of selecting tasks that the GA could be applied to - I'm working on a project to investigate the use of GAs in the desing of communication systems.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R6 : A network of Sparc stations, running UNIX, with self-written software (written in C++, using Sun's compiler).

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R6 : Nothing - this is generally very straightforward.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R6 : It is important to be careful that there are no weaknesses in the evaluation function definition, as the GA has been seen to exploit them. Producing effective evaluation functions is most difficult when a trade-off or compromise is required between a number of system performance measures.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R6 : Early experiments produced a reliable structure for the GA which has been applied without any problems to a variety of applications.

*Q.5.4: SELECTING PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R6 : See previous comment.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R6 : No.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R6 : In some of the cases, the ideal solution is known. The GA has been found to produce close to ideal solutions.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R6 : Haven't bothered.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R6 : Variation between members could be easily observed. Convergence could also be spotted easily.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R6 : Too much information displayed at once could hide useful information.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R6 : Less of an 'information swamp'.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R6 : How does the user define a 'representative' set of chromosomes. They may well NOT be representative at many or all points of a particular run.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R6 : Gives an indication of convergence.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R6 : The fitness vs. generation graph is usually very noisy, particularly with high variations in

the fitness function between good and bad members. The gradient of this curve would have to be averaged to produce a useful value, and the averaging needed may well depend on the particular application.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R6 : For binary strings, it would be possible to determine whether every possible allele existed in the initial population, and how well different alleles propagated.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R6 : For non-binary strings, the number of possible alleles for each gene is likely to be prohibitively high.

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R6 : This could give an indication of whether the mutation rate was too high (interfering with the evolution process) or too low (allowing stagnation). It could also indicate the introduction of a previously unencountered allele on the chromosome.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R6 : None.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED

TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R6 : Possible to observe correct operation of the GA.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R6 : Unnecessary.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R6 : A better indication of convergence than the gradient of the fitness vs. generation graph.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R6 : None.

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R6 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN,

PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R6 : For my use - very limited.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R6 : Useful - if the GA is not converging, altering the mutation rate could help.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R6 : For my use - limited.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R6 : I use steady-state GAs, so the gene-pool is the same as the population.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R6 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R6 :

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R6 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R7*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R7 : AI course Fall 1991 w/Melanie Mitchell. Fascinating. For research since 4/94.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R7 : Computer Architecture/Microprocessor Design.

This is a multi-dimensional combinatorial optimization problem with multiple objectives.

The problem is this: How can I partition millions of transistors into dozens of on-chip hardware structures (memories, adders, etc) to satisfy multiple budget constraints and multiple objectives toward identifying a set of near-optimal partitions? Paper in upcoming 6th ICGA conference.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R7 : I confess that I have always found it fascinating and jumped at the justified chance to play with it in my research.

In addition, it has several characteristics that make it appropriate for my specific design problem.

Among these characteristics:

+ It is readily parallelized on networks of engineering workstations. This is how real-life design engineers work.

+ My objective function is very long (5+ hours) and I need a parallel approach.

+ As a designer, I am looking for sets of near-optimal solutions with which to study and propose the next design improvement. What are the design and performance characteristics of near-optimal solutions, and what should I do next to improve the design further? I am less interested in a single point in the design space.

+ The GA readily handles multiple objectives

+ The GA is a true global search technique.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R7 : Language: C for my GA and objective function/simulators.

Heterogeneous networks of workstations running variants of UNIX.

Primarily, I use DEC workstations (MIPS-based) running ULTRIX. Sun workstations running Sun-OS. I may get involved with HP workstations too, and have already ported the code over to HPs. I use DEC's for development, and Suns for full-blown runs.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R7 : For my problem, this is not too much of a problem. For other problems, this is a major issue, indeed perhaps the single major problem leading to success or failure of the GA in a non-traditional representation domain.

*Q.5.2: PRODUCING AN EFFECTIVE FUNCTION?

R7 : I call a simulator in my objective function. It took me a long time to write this simulator. However, this is independent of the GA; the simulator is hard whether I do hand-optimization or any other global technique. There is nothing GA-specific about the simulator.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R7 : I read the literature and integrated what I learned. I admit this may not be optimal, and knowing what is optimal would be good.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R7 : See 5.3

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R7 : Nothing special. I keep 200 networked workstations busy in parallel. So I take great care to make sure everything is debugged and in-order before I run. But, that is the nature of research, eh?

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R7 : Repeatability is a major way for me to know the GA is not simply hacking about. If I can get *nearly* the same design multiple times, I am confident that the stochastic optimization is OK. Also, I sanity check, and plot the gene and objective values as the simulations proceed. Premature convergence has been the biggest GA problem I have had to address. Once that was solved, things

seem to be pretty good.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R7: See 6.1

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R7: Fun. Good to teach non-believers. Could get an early-on sense of what was happening. I do this already by making plots and it is semi-automated. As such, I expect that other serious GA researchers do the same.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R7: I can imagine "visualizing" real-valued parameters. But how to you visualize other problem-specific genetic representations, e.g., tree-based, etc.? Is a general-purpose display method even possible in consideration of the number of possible genetic representations?

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R7:

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R7:

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R7: Yes since this tells the user s/he may be nearly done, or at least near a plateau. Also, I watch the standard deviation of objective values.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R7:

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R7: Good. I do this already in a non-fancy semi-automated way so I can watch what design is evolving as it proceeds. My entire GA run might take 2+ weeks so I need to watch for sanity as it proceeds.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R7:

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R7:

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R7:

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R7: Fun, good teaching/debuggin tool. I am not sure how to digest and exploit this info, however, over 1000s of runs in a real application.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R7:

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R7: Yes, many.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R7:

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R7:

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTER-

ACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R7: Again, fun, and good for teaching but that is all.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R7: Again, fun, and good for teaching. I suppose it could be used "heuristically" to change direction of search, but this is an ad-hoc approach to an already stochastic optimization technique.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R7: Again, fun, and good for teaching. However, probably disruptive to the GA. Hard to say.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R7: Again, fun, and good for teaching. However, probably disruptive to the GA. Hard to say.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R7:

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R7:

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE

RESULTING GA VISUALIZATION SYSTEMS?

R7: No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *R8*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

R8 : 4 or 5 years.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

R8 : Algorithm optimisation, curve fitting. Have also done basic work on GAs themselves.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

R8 : GAs offer a general method for solving optimisation problems; also because of interest in the GAs themselves.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

R8 : Various - hard to be specific.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

R8 : One of the hardest problems, if not the hardest.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

R8 : Hard for combinatorial problems rather than function optimisation; often depends on 5.1 in

such cases.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

R8 : The operators depend very heavily on the representation.

Initial population is fairly unimportant.

Reproduction method is somewhat important, but not the biggest issue.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

R8 : Not too hard. The trouble is that sometimes, larger populations were needed than could realistically be used...

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

R8 : Choosing when to halt the GA is another problem.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

R8 : In most cases, the fitness function itself was used; in the curve case, visual inspection was also useful.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

R8 : I hope it isn't! I just want it to explore the "right part" of the space. More seriously, multiple

runs were used to see how often the same solutions resulted.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

R8 :

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

R8 : Far too many for this to be useful.

The chromosomes themselves are not very meaningful in some of our work - ie, complex calculations are needed to derive the individuals from them.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

R8 :

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

R8 : Same.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

R8 :

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

R8 :

Not a lot, because the fitness versus generation graph is quite noisy, and so derivatives would be measuring noise.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

R8 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

R8 : See above.

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

R8 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

R8 : See above.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

R8 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

R8 : See above.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

R8 :

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

R8 : Far too simple a measure of similarity.

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

R8 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

R8 : Useful for debugging, but not in production runs.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

R8 : Maybe useful if the GA gets stuck.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

R8 : See earlier remarks.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

R8 : See earlier remarks.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

R8 : Ability to introduce a strong mutation pulse to kick the GA into a different region of solution space.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

R8 : The really tricky issue is designing the representation and the operators, not controlling or visualizing the GA during running.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

R8 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - *A1*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A1 : For almost a year.

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A1 : Optimization of designs in the mechanical engineering field (and optimization of some test problems).

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A1 : Previously, a Monte Carlo method was used. We found out the GA was much more effective at solving large problems.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A1 : We wrote our own GA programs. Based on the first Fortran program on an MS-DOS compatible that used the Monte Carlo method, a GA was developed for that environment. Lateron, a version was written in C. It was a more powerful and flexible version and it was developed on an Amiga. It is written in ANSI C and is therefore portable.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A1 : When solving mechanical engineering "optimization" problems, the strings are the collection of independent design variables, so this mapping is no problem at all.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A1 : That can be quite difficult. It would go too far to go into detail about the problems that can arise, but there are problems in the field of multi-variable fitness evaluation. Furthermore, in the field of mechanical engineering, criteria are often determined by the "contractor" and the importance of these criteria are not as "fixed" as you would like.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A1 : The ANSI C version of the program we developed can use multiple mutation and crossover methods. These can be selected at runtime and turned on and off while the algorithm is running. This gives the user the ability to experiment with the different methods and to gain more insight into them.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A1 : Population size is chosen rather arbitrarily at the moment (limited by memory and practical speed limitations). Mutation rate as well as crossover rate can be adjusted at runtime (constantly).

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A1 : We did design some preprocessors to make problem definitions easier, but these have nothing to do with the GA. So the answer is: no.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A1 : In mechanical engineering, it is quite easy to check the results against existing designs. We have done this in some cases.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A1 : We didn't do extensive research on this, because we got satisfying results, which indicated that the problem space was searched quite well.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A1 : A detailed "report" of the current population, providing a lot of data to those that can "read" it correctly.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A1 : Might confuse people, and might tempt people to draw the wrong conclusions. I mean, the GA is strongly stochastic, so one must always be careful about drawing conclusions from any particular run.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A1 : Would give even better controlled data. This would definitely be better than 7.1.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A1 : Same disadvantages, with the additional risk of accidentally disregarding data that might be important after all.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A1 : Not sure about these...

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A1 : Not sure about these...

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A1 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A1 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A1 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A1 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A1 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A1 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A1 : All these statistics would give more insight into the GA, and would therefore be quite nice for educational purposes.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A1 : It slows down the GA and therefore (in my particular application) the optimization process, which is a disadvantage (that might or might not be important, depending on your goals, computer speed, etc.).

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A1 : In the field of optimization, the independent variables are also displayed (at least, we display them). In general, one could perhaps say that the "data that the chromosomes represent" should be shown (optionally).

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A1 : We have already implemented "start" and "stop" and interactive changing of the parameters.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A1 : This can be very useful, to speed up the algorithm and to increase the user's insight into the process.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A1 : I don't really see the use of this. But perhaps I'm missing something here. It wouldn't hurt as an option, that's for sure.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A1 : Same as 10.3, don't see the use at the moment.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A1 : Nothing comes to mind at the moment.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE

EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A1 : I am still learning more about GA's every day. In the field of mechanical engineering, we have used them to optimize designs. I keep talking about "we", and I think I should explain myself. I am studying mechanical engineering at the Delft University of Technology and "we" refers to my professor and other people at the department of mechanical engineering design.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A1 : No. I would not object to being contacted in the future.

I'd be happy to further discuss aspects of GA's and the visualization of them. I can send screenshots of my program (that is the ANSI C program, which I wrote) that show how I visualized everything. Of course, this is only an example. I know there is room for improvement. I am therefore interested in any new suggestions you might have on this subject.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - A2

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A2 : 18 months

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A2 : I am using GAs for the design of Predictive Controllers.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A2 : Because classical methods of optimization cannot solve the problem mentioned above.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A2 : Currently I am programming in Borland C++ v.4 for DOS. After next month I will start to use Linux OS and gcc.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A2 : Not difficult

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A2 : Not difficult

*Q.5.3: CHOOSING THE GA's COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A2 : Not difficult but complicated

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A2 : complicated

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A2 : You have mentioned everything

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A2 :

1. Extend searching
2. Quality of the solutions (final performance)

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A2 : The main point is the final performance considering at the same time the current practical issues.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRE-

SENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A2 : Supervisory control to all the individuals.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A2 :

1. For large populations?
2. It's difficult to check all the candidates

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A2 : Ability for someone to be experimented, so that to choose the best possible representation for the specific problem.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A2 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A2 :

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A2 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A2 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A2 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A2 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A2 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A2 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A2 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A2 :

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A2 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A2 : You have not left anything for me to think. The questions mentioned above, definitely have only advantages.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A2 : HELPFUL

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A2 : HELPFUL

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A2 : HELPFUL

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A2 : HELPFUL

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A2 : You have thought everything.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A2 : The transfer of the biological terminology to GAs field must be more direct and more conceivable.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A2 : Yes. I would object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - A3

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A3 : two years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A3 : for biological applications: aligning protein sequences, folding RNA molecules, finding the best set of parameters for a specific application.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A3 : the problem of multiple sequence alignment with the 'sums of pairs' used as an objective function is known to be NPcomplete. Thus, as the problem can be approached in a combinatorial way, it looked like a good idea.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A3 : UNIX,C & VMS, C

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A3 : I found quite generally that the 'naive' approach rarely works. Thus, the mapping seems to me the most crucial point in the strategy of designing a GA

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A3 : In my case, the evaluation function already exists, so most of the time there is no real choice.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A3 : I found that a lots of changes in the selection scheme, generation production and other have drastic effects. But again, this is very difficult to control. I am now working with a model using most of the features described by DAVIS in 'The handbook of GA'.

This is not necessarily the best model, but it works reasonably well, and because of the fuzziness around these parameter I am less and less keen on playing with them. It seems to me much more worth spending time on the quality of the mapping and the quality of the operators.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A3 : It seems to me that the population size is not a real problem. In my experience, GA are quite robusts regarding this parameter. The population size may be GA/problem specific, but for a given class of problem in a given GA, using always the same pop size does not seem to be a problem.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS TAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A3 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A3 : We use as a benchmark, an exhaustive program that can provide a guaranteed optimal solution for a small problem.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A3 :

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A3 : none

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A3 :

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A3 : if properly done, it could help visualising the emergence of some niche, and maybe their relations

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A3 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A3 :

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A3 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A3 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A3 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A3 : allow the the user to really get a feeling of what the mutation does, and possibly, what are its limits.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A3 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO

CHROMOSOMES BY A SINGLE POINT Crossover OPERATOR - ADVANTAGES.

A3 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND Crossover BETWEEN TWO CHROMOSOMES BY A SINGLE POINT Crossover OPERATOR - DISADVANTAGES.

A3 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A3 : this might be quite usefull in helping to identify problems that need a 'niche' approach

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A3 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A3 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECU-

TION:

A3 : This would be extremely usefull

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A3 : This probably depends on the type of problems to solve, It would probably help for large problem that only need to be solved once

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A3 : no

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A3 : no

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A3 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A3 :

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A3 : yes

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - *A4*

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A4 : 2-3 years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A4 : as a search algorithm and as an gave development system

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A4 : interest, evidence that GA work well as search algorithms

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A4 : Linux boxes, SGI, Cray supercomputers, all in C/C++ (GNU)

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A4 : Doing analysis of proteins, very simple mapping

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A4 : Hard to describe what a "GOOD" protein is, but thats a problem with the field not with GA

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A4 :

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A4 : Hard to find good parameters, did mostly trial and error, still searching for good parameters

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A4 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A4 :

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A4 :

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A4 :

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A4 :

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A4 :

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A4 : variability of fitness criterion (may be an advantage truthfully)

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A4 :

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A4 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A4 : quick analysis of relationships genes

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A4 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A4 : again gives you an overall picture of gene changes

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A4 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A4 : Very interesting, good way to actually see if what you plan is actually working

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A4 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A4 :

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A4 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A4 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAS?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A4 :

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A4 :

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A4 :

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A4 :

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD

CONSIDER BENEFICIAL.

A4 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A4 :

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A4 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - A5

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A5 : since 1992, approx 3 yrs

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A5 : optimisation, adaptive search to identify design options, integration with NN.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A5 : optimiser, good search tool

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A5 : Sun Sparc Stations, C lang, POP11 lang

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A5 :

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A5 :

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A5 :

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A5 : yes

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A5 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A5 : constraint satisfaction

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A5 : heuristics, otherwise difficult

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A5 : all info

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A5 : too much info, unnecessary

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A5 : helpful

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A5 : there can be a chance to loose novel chromosome structure.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A5 : OK, can give some idea about convergence

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A5 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A5 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A5 : not much info

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A5 :

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A5 : not much info

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A5 :

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A5 : will not be meaningful in multidimensional problem situation.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCP") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A5 :

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A5 : that is not representative of binary representation

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A5 : best solution achieved every generation

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A5 : would be very good

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A5 : not very good idea, instead an adaptation scheme can be developed.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A5 : not good idea, that would interfere in GA's search strategy.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A5 : not good idea.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD

CONSIDER BENEFICIAL.

A5 : initial partial seeding of population with some "good" chromosomes (using domain knowledge).

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A5 : some hybrid approach could be useful, like for few generations if GA can not find any improvement in terms of fitness, then may be hillclimbing can be started from that point or even simulated annealing or tabu search.

I am developing Adaptive Search Manager using Fuzzy Expert Systems, which is expected to extract info from GA search and utilise that info/knowledge for effective search.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A5 : No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,
The Open University, Milton Keynes MK7 6AA.

Respondent - A6

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A6 : 4 years on and off (last 2 continuously)

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A6 : 1. Process Planning

2. Mechanical Design

3. Mechanical Durability Assessment Test setup procedure

Number 3. is my current topic

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A6 : 1. Extension of previous research of another.

2. Feasibility study leading to full research project by another. Expected that GAs would mimic the method nature uses for design.

3. It was expected that GAs would deal efficiently with the large data sample that exist in simulation testing. Also, since we only have to convert data in a forwards direction, the existing problems with methods in current use would not be encountered.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYS-

TEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A6 : HP workstation Mathworks MATLAB

It is expected that the development will also work using MATLAB on an PC.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A6 : The initial creation of the population would produce many unfeasible solutions. Development of the representation method has (hopefully) solved this.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A6 : Was initially a problem to define one which gave good solutions enough of an advantage over weaker members, but which did not completely exclude these members.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A6 : Initial population (see 5.1 above)

Operators etc. Choice not really a problem. The parameters used with them, however, make a large difference to results and solution time.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A6 : See 5.3

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A6 : No GA related ones.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A6 : The evaluation of my problem gives a percentage match.

Therefore, a match of 100% is a perfect solution.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A6 :

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A6 : Good for investigation as to what the GA is doing.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A6 : Far too much data for me, since I am looking at an application rather than the GA itself.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A6 : Better than 7.1, giving some info on the current generation.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A6 :

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A6 : Gives info on how the search is proceeding

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A6 :

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A6 :

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A6 :

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION

OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A6 : Good for following what has been happening, therefore the understanding of what is going on.

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A6 :

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A6 : Same as above.

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A6 :

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCI") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A6 : Would allow knowledge of the diversity of the population. This would allow, through experimentation on your problem knowledge as to how the search was likely to proceed.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCI") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A6 :

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A6 :

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A6 : Very helpful (a definite)

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A6 : Could be quite useful for me as I will also be using a GA to identify the mechanical system. If I obtain a 'better' parameter set for my dynamic system whilst the main GA is running it would be useful to be able to introduce this to the current run.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A6 : It may be useful to be able to input new chromosomes during the run to allow for expert knowledge to be incorporated.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A6 : see above.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A6 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A6 :

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A6 : no

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - A7

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A7 : About 4 years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A7 : Studying the optimum structure of the Australian sheep breeding industry.

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A7 : There are a large number of variable option, all of which interact. For example - the number of rams and ewes held in nucleus groups, the number of years these rams and ewes are used before replacement, the number of rams available for commercial flocks, the number of good quality ewes being promoted into the nucleus, the selection methods used for all of these sheep and the use of artificial insemination and multiple ovulation.

In most of these cases some intermediate value is optimal (e.g. more expensive methods of selection are more accurate, but the cost tends to increase exponentially for only small gains in accuracy), and the optimal values depend on choices for other components in the system. I have selected 17 variable items for use in my main GA. I have used GAs for other specific aspects of the breeding system, but the answers here relate to my main system.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A7 : Borland Pascal (not Windows version) on an IBM compatible 486DX 33Mhz. However, I borrow a Pentium whenever possible.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A7 : Most variables I use are continuous, so I have to map them to a series of integers - usually 16 or 32 to avoid too many large genes. This makes the solution 'lumpy', so I sometimes fine tune it in a narrow range after first finding the appropriate range with several broad scale runs. Several of my variables are percentages and these do not map well to powers of 2.

An early problem was the tendency of many strategies to produce impossible results. For example a common problem was the inability of one of the breeding groups to maintain its population because the ewes were moved out before they could produce enough replacements. In other circumstances (e.g. multiple ovulation) the same strategy might be a winner. I finally fixed this by trying to ensure that the genes would produce a legal result. In the above case by setting the gene to determine the number of EXCESS lambs produced after satisfying the minum requirements, rather than the ACTUAL number of lambs. This requires reordering the calculation, but always gives a valid result.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A7 : This is by far the biggest job, as the theoretical genetics are very complex (for me anyway). The evaluation section requires about 80kb of code and takes about 1 second per evaluation on my 486DX33. I have no idea how this type of evaluation could be incorporated into a genral purpose GA program, except as unit to be compiled with other GA specific units.

*Q.5.3: CHOOSING THE GA's COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A7 : I worked through Goldberg's book, using his simple GAs in Pascal, then modified the programs

taking into account his comments on potential improvements and any ideas that came to me at the time, with allowance for my specific problems, but keeping the GA section general enough to apply to any other problems.

The main change from Goldberg's simple GA is that when I normalize the fitness function, I set the average value to exactly 1.0, the minimum to zero, and the current maximum to 2.0. This requires separate linear scaling for those above zero, and those below zero. It avoids the problem of a few extremely good or extremely bad values skewing the whole distribution. I have been told that ranking would do this better, but do not know a fast ranking method that would make any further gains worthwhile.

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A7 : I could not find any really good guidance here, so I have experimented a bit. I found that the mutation and crossover rates did not make a big difference, so settled on intermediate values that seemed satisfactory for my main function. Borrowing from nature, I use a circular chromosome (i.e. there is always an even number of crossovers). I have done some further experiments now that I have access to a Pentium and am considering options like variable crossover and mutation rates. At present I usually use 4-10 crossover sites per pair, and have about 20% of the population as mutants. However, I do not know of any good methods of selecting these other than trial and error.

My usual population size is 300, which I understand is rather large for a 67 bit chromosome. There is only slight improvement in results compared with 100 or 200, but the smaller populations have definite tendency to sometimes arrive at a suboptimal level, apparently due to inbreeding and loss of specific bits in the early stages. I am rather sensitive to inbreeding as it plays a critical role in my own sheep breeding structure, so I favour large populations even if it takes longer to get results.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A7 : I have a file of default settings for all the GA settings and those specific to the problem. These defaults can be changed if necessary. Apart from population size, mutation rate and crossover rate, I have options to switch Gray codes on/off and allow or disallow clones (identical chromosomes). I also set an upper limit to the number of generations (normally 120) for when I run a series of tests overnight.

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A7 : I normally test any problem 5-10 times, compare it with hill-climbing results and also use my own intuition to try obvious solutions to find out why the GA didn't use them. It is rare for the GA not to find the best possible solution in 10 tries. Hill-climbing can get good solutions, but I have found that it never gets the best solution unless the starting conditions are artificially set with extreme values for some variables. My intuition never gets the best solution, but can be used as a starting point for hill-climbing to reach the best. I also look at the intermediate calculations used in the optimum to check that they make biological sense.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A7 : I rely on my intuition and knowledge of the subject to check any solutions that do not appear to be produced by the GA. Usually by editing to create a test subject, then hill-climbing it on the other genes.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A7 : I did this when first starting simple test functions, and it did help to verifying that my program was doing the right things.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A7 : I quickly stopped looking at each individual as it is too confusing and not informative.

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A7 : My current system shows the 17 gene values for the best 5 individuals. This gives me some idea how things are going, whether they are converging and which genes are still highly variable.

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A7 : Although the best 5 are always shown I usually only look at the best one (shown in a different colour) and use the current minimum, average and maximum to check how the GA is going.

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A7 : When I do many overnight runs to test the settings I compare them using a graph of fitness versus generation. This show how quickly different settings reach good values, and how close they get to the highest possible value. This is useful because some settings make the best gains early, but seem to run out of variation and fail to reach the maximum that slower settings can reach. I have to do this manually in Excel form test files produced during the run.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE

GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A7 : I do not find this useful for examining the actual results (i.e. in terms of my sheep breeding system) except to get some idea whether further increases might be possible if left for more generations.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A7 : This is only useful when I have made changes to the program in this section and need to check that I have not introduced a new bug.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A7 : No need unless the program is not working correctly.

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A7 : Same as 8.1

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A7 : Same as 8.1

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A7 : Same as 8.1

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A7 : Same as 8.1

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A7 : There might be some value in a convergence value that summarised the whole population so you could check the rate of convergence and decide when no further gains were likely.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE IF ITS BIT POSITIONS ("LOCT") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A7 : Doing this for individual chromosomes would be confusing.

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A7 : I show either of two 'graphs' during the run. These are done using the standard ASCII block graphic characters in four colours to give about 16 levels of colour/shading, from full red for high values to full blue for the lowest.

One shows the total number of positive bits in each gene over the whole population. This show me which genes have fully converged and which bits still have high variation.

The other 'graph' shows the actual gene values (ranged from minimum to maximum) to show which values are being favoured and which values are dropping out.

These graphs allow me to glance at the screen and decide whether it is worth stopping, or whether it should run a bit longer. The latter also indicates whether some values are still at a high level even if not present in the top 5 shown individually.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A7 : I find it valuable to be able to stop at the end of any generation, then look at the details of the current maximum, step forward, or continue running. I can save the current complete set when paused and often save intermediate stages in important runs. I have never felt any need to step back to a previous generation. I have an option to store the best individual from every generation in a file, so that I can view the whole run and see which genes stabilised early, and which settled down later in the run.

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A7 : I can do this at any time when paused, (and I can only pause between generations), although I rarely do so, except sometimes to lower the population size if I am feeling impatient. I never change the evaluation settings during a run as there is no point (in my case) in running a GA in a changing environment.

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A7 : I find it very useful to be able to edit the chromosome. This is often done to compare my intuition with the current settings, or to check whether small variations in the current optimum would further improve it. More usually I find out why my intuition would give a worse

answer. In some cases if my graphs indicate that certain values are not being used I can seed the population with an individual with these values and see if can spread these genes in future generations.

I do the editing by using the current best chromosome as a default, then the edited chromosome replaces the current worst individual in the population.

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A7: I only edit when paused at the end of a generation. I can't think of any reason to stop during a generation.

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A7: In order to evaluate why a particular individual is the best (compared with my own ideas) I have a full 50 line screen of data showing intermediate calculations used in the evaluation function. This is essential to determine the effect of minor (and major) changes to the current settings, to show the effect of each gene in the whole picture, and to allow me to explain why the best individual is better than other alternatives.

This type of display is obviously specific to any particular problem. However, any program with an evaluation function should be able to show specified intermediate calculations in that function.

While in the edit mode I have the option to look at any single gene, or any pair of genes to see what values occur with changes over the full range of these genes (with all other genes held constant). This helps to check how much influence a given gene has on the current system, as well as checking whether it is at the true optimum. The 2-gene system is particularly useful here, but I can't think of a good way of showing 3 or more genes at once.

The above display can either show the actual values, or use a 16 shade graph as described previously.

The 2-gene graph often shows diagonal ridges, where changing any single gene gives a worse rather than better result, whereas changing both genes can lead up the diagonal ridge to better values. I presume the same diagonal ridges occur in higher dimensions.

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A7: It is important not to get locked into using GAs for problems where simpler (faster) methods will do as well or better. It is also important to be satisfied that a GA is best for some problems. I have included the option to carry out several varieties of hill-climbing (and simulated annealing), and this has convinced me to stick with GAs as the main method.

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A7: No. I would not object to being contacted in the future.

GA Visualization, Design Questionnaire.

Trevor Collins, The Knowledge Media Institute,

The Open University, Milton Keynes MK7 6AA.

Respondent - A8

*Q.1: HOW LONG HAVE YOU BEEN USING GAs?

A8 : About 7 years

*Q.2: DURING THIS TIME WHAT HAVE YOU USED GAs FOR?

A8 : A variety of scientific problems

*Q.3: WHY DID YOU USE GAs FOR THIS TASK?

A8 : They seemed to offer the prospect of providing better results, or equivalent results in less time, than conventional techniques.

*Q.4: WHAT ENVIRONMENT(S) DO YOU USE WHEN WORKING WITH GAs? PLEASE SPECIFY EACH COMPUTING ENVIRONMENT SEPARATELY I.E. THE COMPUTER SYSTEM, PROGRAMMING LANGUAGE AND/OR APPLICATION TOOL?

A8 : Genarally HP 9000 workstations running HP-UX 9.01, programming in C. We do not use commercial or shareware packages, but write all our own software.

*Q.5: WHAT DO YOU FIND DIFFICULT, IF ANYTHING, ABOUT THE FOLLOWING SET-UP STEPS INVOLVED IN CREATING A GA:

*Q.5.1: DEFINING THE MAPPING BETWEEN THE PROBLEM DOMAIN AND THE STRING REPRESENTATION USE BY THE GA?

A8 : This is not necessarily difficult, but clearly important. We have several times ued multi-dimensional GA strings (on studies of the movement of air pollution and, more recently, studiss on the analysis of liquid waste) since these provide better results for certain types of problem. We often

use non-standard coding.

*Q.5.2: PRODUCING AN EFFECTIVE EVALUATION FUNCTION?

A8 : This 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.

*Q.5.3: CHOOSING THE GA'S COMPONENTS, E.G. THE INITIAL POPULATION CREATION METHOD, WHAT REPRODUCTION GENE-POOL SELECTION CRITERION TO ADOPT, WHICH GENETIC OPERATORS TO APPLY, ETC.?

A8 : A bit of trial and error is often required. One becomes more familiar with certain strategies, and I suppose one tends to favour those strategies, perhaps unreasonably, over others. Members of my group have a pretty free hand, and are usually eager to investigate any different approaches they can find, and not be guided much by my own experience!

*Q.5.4: SELECTING SUITABLE PARAMETERS FOR THE GA, E.G. THE POPULATION SIZE, THE MUTATION RATE (IF APPROPRIATE), ETC.?

A8 : Trial and error, starting from parameters which past experience suggests will be productive.

*Q.5.5: ARE THERE ANY OTHER SET-UP STEPS THAT YOU USE BEFORE RUNNING THE GA? IF SO PLEASE NOTE THEM AND ANY ASSOCIATED DIFFICULTIES YOU ENCOUNTER BELOW.

A8 :

*Q.6: HAVING APPLIED A GA TO A PARTICULAR PROBLEM WHAT APPROACH DO YOU TAKE, IN ORDER TO:

*Q.6.1: ASSESS THE QUALITY OF ANY SOLUTION(S) FOUND?

A8 : Comparison with literature results if available. Comparison with results yielded by conventional approaches on the same data. Statistical analysis of the results yielded by the GA. Comparison between results of repeated runs.

*Q.6.2: EXAMINE HOW REPRESENTATIVE THE OUTPUT OF THE GA IS IN TERMS OF ALL THE POSSIBLE POINTS WITHIN THE PROBLEM-SPACE?

A8 : Repeated runs. Statistical analysis of runs. Investigation of the surface through gradient search and other local search techniques. Visualisation of the surface. Comparison with random search results. Theoretical methods if available.

*Q.7: IF THE FOLLOWING TYPICAL OUTPUT CHARACTERISTICS WERE TO BE REPRESENTED WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.7.1.A: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - ADVANTAGES.

A8 : None, unless the population were very small. It is often useful to have a measure of the diversity of the population, but one (or several) numerical values representing this would be preferable in most instances to viewing data on 50 or 100 individual strings.

*Q.7.1.D: ALL OF THE INDIVIDUAL CHROMOSOMES WITHIN EACH POPULATION - DISADVANTAGES.

A8 : Too much screen clutter

*Q.7.2.A: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - ADVANTAGES.

A8 : Better. Less screen clutter

*Q.7.2.D: A USER DEFINED SELECTION OF REPRESENTATIVE CHROMOSOMES - DISADVANTAGES.

A8 : None

*Q.7.3.A: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - ADVANTAGES.

A8 : A standard method of following the progress of the calculation. Generally gives a useful idea of how things are going.

*Q.7.3.D: THE RATE OF CHANGE IN THE POPULATIONS FITNESS VALUES, I.E. THE GRADIENT VALUES OF A FITNESS VERSUS GENERATION GRAPH - DISADVANTAGES.

A8 : One often wants a more detailed understanding of what is happening in the population than this graph can give.

*Q.8: AS WELL AS DIRECTLY ILLUSTRATING THE OUTPUT OF THE GA, VISUALIZATION COULD BE USED TO REPRESENT ADDITIONAL INFORMATION EITHER DERIVED FROM THE OUTPUT DATASET OR RECORDED SEPARATELY. IF VISUALIZATION WERE USED TO REPRESENT THE FOLLOWING CHARACTERISTICS WHAT ADVANTAGES OR DISADVANTAGES, IF ANY, COULD YOU FORESEE?

*Q.8.1.A: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - ADVANTAGES.

A8 : Depends upon the problem being tackled. We have found such a visualization useful at times.

*Q.8.1.D: THE CHROMOSOMES IN THE REPRODUCTIVE GENE-POOL - DISADVANTAGES.

A8 : None

*Q.8.2.A: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION OPERATOR HAS BEEN APPLIED - ADVANTAGES.

A8 : limited

*Q.8.2.D: THE OCCURRENCE OF MUTATION IN CHROMOSOMES WHERE A MUTATION

OPERATOR HAS BEEN APPLIED - DISADVANTAGES.

A8 : Since mutation normally causes little change in the string, there wouldn't be a great deal to show! There should be no value in showing the position of mutation, unless for some reason one biases the position. I can't see this being very useful.

*Q.8.3.A: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - ADVANTAGES.

A8 : limited

*Q.8.3.D: THE INTERNAL ACTIONS OF THE GENETIC OPERATORS BEING APPLIED TO THE CHROMOSOMES, E.G. THE SPLITTING AND CROSSOVER BETWEEN TWO CHROMOSOMES BY A SINGLE POINT CROSSOVER OPERATOR - DISADVANTAGES.

A8 : Again this would be of interest in illustrating how the GA works, but I think of little value in helping one monitor the action of the algorithm.

*Q.8.4.A: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - ADVANTAGES.

A8 : We've used this type of measure a lot. Useful.

*Q.8.4.D: A "SIMILARITY" RATING FOR EACH CHROMOSOME BASED ON HOW LITTLE THEY DIFFERED TO THE FITTEST CHROMOSOME, E.G. A TEN BIT BINARY CHROMOSOME THAT DIFFERED FROM THE FITTEST CHROMOSOME IN THREE OF ITS BIT POSITIONS ("LOCF") MAY HAVE A SIMILARITY RATING OF 0.7 - DISADVANTAGES.

A8 : None

*Q.9: PLEASE SPECIFY ANY OTHER DIRECT OR INDIRECT CHARACTERISTICS THAT

YOU WOULD BE INTERESTED IN SEEING VISUALIZED.

A8 : Varies greatly from one application to the next. The most useful factors we follow relate to the degree of diversity within the population.

*Q.10: HOW HELPFUL, OR DESTRUCTIVE, WOULD YOU FIND THE FOLLOWING INTERACTION OPPORTUNITIES FOR YOUR USE OF GAs?

*Q.10.1: EXECUTION CONTROL THROUGH THE USE OF A CONTROL PANEL TO RUN, PAUSE STEP FORWARD, STEP BACKWARD, SAVE A SNAPSHOT, AND/OR STOP EXECUTION:

A8 : useful

*Q.10.2: EDITING THE ALGORITHM'S PARAMETERS DURING EXECUTION:

A8 : of some interest

*Q.10.3: EDITING THE POPULATION'S CHROMOSOMES BETWEEN TWO GENERATIONS:

A8 : of minor value

*Q.10.4: EDITING THE REPRODUCTION GENE-POOL'S CHROMOSOMES WITHIN A GENERATION:

A8 : of minor value

*Q.11: PLEASE SPECIFY ANY OTHER FORMS OF INTERACTION THAT YOU WOULD CONSIDER BENEFICIAL.

A8 :

*Q.12: DO YOU HAVE ANY OTHER SUGGESTIONS ON HOW GAs COULD BE MADE EASIER TO USE? OR ANY OTHER COMMENTS AT ALL ABOUT GAs? PLEASE NOTE THEM BELOW.

A8 :

*Q.13: FINALLY, WOULD YOU HAVE ANY OBJECTION TO BEING CONTACTED IN THE FUTURE WITH REFERENCE TO THIS PROJECT AND THE EVALUATION OF THE RESULTING GA VISUALIZATION SYSTEMS?

A8 : No. I would not object to being contacted in the future.

Appendix D

GONZO Example Applications

This Appendix contains the Lisp code used in Gonzo to produce the examples described in Chapter 7, Section 7.3. The code used in GECO to produce online visualizations is given in section D.1. The code used in GONZO to produce the offline visualizations of the maximum integer problem, the De Jong F1 test problem, and the royal road function, are given in Section D.2.

D.1 Online Visualization

The following annotated version of the GECO EVOLVE method is used to produce online visualizations in GONZO. The annotations made to the original GECO EVOLVE method are shown here in bold.

```
(defmethod EVOLVE ((self ecosystem))  
  (unless viz::*visualization-dialog* (viz::create-visualizations self))  
  (evaluate self (plan self))  
  (mapcar #'(lambda (view)  
    (setf (current-generation-range view)  
          (mapcar #'incf (current-generation-range view)))  
    (setf (total-generation-range view)  
          (list (first (total-generation-range view))  
                (incf (second (total-generation-range view))))))
```

```

(cond ((or (< (min-score (elt 0 (statistics (plan self))))
            (first (total-fitness-range view)))
        (> (max-score (elt 0 (statistics (plan self))))
            (second (total-fitness-range view))))
      (setf (total-fitness-range view)
            (list (min (min-score (elt 0 (statistics (plan self))))
                    (first (total-fitness-range view)))
                  (max (max-score (elt 0 (statistics (plan self))))
                    (second (total-fitness-range view)))))))
      (viz::views viz::*visualization-dialog*))
(unless (evolution-termination-p (plan self))
  (incf (generation-number self))
  (regenerate (plan self) self)
  (evolve self)))

```

D.2 GONZO Example Problem Visualizations

This section presents the code used in GONZO to produce the three example problem visualizations presented in Section 7.4.

D.2.1 The Maximum Integer Problem

The following Lisp code was used to produce the GONZO visualization shown in Figure 7.9 of the maximum integer problem, see page 199.

```

(defvar *visualization-dialog* nil) ;; visualization container dialog

(defun maxint ()
  (test-plan 'run-1 1 'maxint-plan) ;; GECO GA dataset run-1
  (create-visualizations run-1) ;; GONZO create visualizations function

```

```

) ;; test

(defmethod create-visualizations ((run ecosystem))

  (setf *visualization-dialog*

    (open-dialog

      nil ;; list-of-dialog-items

      'my-navigator ;; device

      cg:*screen* ;; stream

      :name 'visualizer ;; name

      :pop-up-p nil ;; not a pop-up dialog

      :background-color cg:white ;; background colour

      :window-exterior (cg:make-box 30 50 1030 850) ;; window exterior box

      :title "Test")) ;; title string for window

  (create-fitness-versus-time-graph

    'fitness-graph-0 ;; name

    run ;; dataset

    *visualization-dialog* ;; parent-dialog

    (cg:make-box 400 600 1000 800)) ;; exterior-box

  (create-fine-grained-chromosome-view

    'text-view-0 ;; name

    *visualization-dialog* ;; parent-dialog

    (cg:make-box 0 350 400 800)) ;; exterior-box

  (create-search-space-visualization

    'scatterplot-view-0 ;; name

    run ;; dataset

    'GSM-D-circle ;; chromosome-mapping-technique

    *visualization-dialog* ;; parent-dialog

    (cg:make-box 400 0 1000 600)) ;; exterior-box

```

```

'D-GSM ;; coordinate-mapping-technique

(list text-view-0) ;; list-of-views

(create-movie-player
  'control-panel ;; name
  '(i< << <1 > 1> >> >i) ;; list-of-lables
  '(start rewind back1 play-pause forward1 fforward end) ;; list-of-functions
  (list scatterplot-view-0) ;; list-of-views
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 0 0 400 85)) ;; exterior-box

(create-generation-fitness-selector
  'view-range-window ;; name
  (list scatterplot-view-0) ;; list-of-views
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 0 85 400 250)) ;; exterior-box

(create-schema-highlight-selector
  'schema-editor-window ;; name
  (list scatterplot-view-0) ;; list-of-views
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 0 250 400 350)) ;; exterior-box

) ;; create-visualizations

```

D.2.2 The De Jong F1 Test Problem

The following code was used to produce the example visualizations of a GA solving De Jong's F1 test problem, as shown in Figure 7.12, see page 202. This code is virtually identical to that used to produce the visualizations of the maximum integer problem given in the previous subsection, the only differences being a change in the GA's genetic plan, the image mapping used in the search

space visualization, and the window dimensions of the schema highlight selector and fine grained chromosome view.

```
(defvar *visualization-dialog* nil) ;; visualization container dialog

(defun dejong ()
  (test-plan 'run-1 1 'dejong-plan) ;; GECO GA dataset run-1
  (create-visualizations run-1) ;; GONZO create visualizations function
) ;; test

(defmethod create-visualizations ((run ecosystem))
  (setf *visualization-dialog*
    (open-dialog
      nil ;; list-of-dialog-items
      'my-navigator ;; device
      cg:*screen* ;; stream
      :name 'visualizer ;; name
      :pop-up-p nil ;; not a pop-up dialog
      :background-color cg:white ;; background colour
      :window-exterior (cg:make-box 30 50 1030 850) ;; window exterior box
      :title "Test")) ;; title string for window

  (create-fitness-versus-time-graph
    'fitness-graph-0 ;; name
    run ;; dataset
    *visualization-dialog* ;; parent-dialog
    (cg:make-box 400 600 1000 800)) ;; exterior-box

  (create-fine-grained-chromosome-view
    'text-view-0 ;; name
    *visualization-dialog* ;; parent-dialog
```

```
(cg:make-box 0 430 400 800)) ;; exterior-box

(create-search-space-visualization
 'scatterplot-view-0 ;; name
 run ;; dataset
 'GSM-D-colour-rectangle ;; chromosome-mapping-technique
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 400 0 1000 600) ;; exterior-box
 'D-GSM ;; coordinate-mapping-technique
 (list text-view-0)) ;; list-of-views

(create-movie-player
 'control-panel ;; name
 '(i< << <1 > 1> >> >i) ;; list-of-lables
 '(start rewind back1 play-pause forward1 fforward end) ;; list-of-functions
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 0 400 85)) ;; exterior-box

(create-generation-fitness-selector
 'view-range-window ;; name
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 85 400 250)) ;; exterior-box

(create-schema-highlight-selector
 'schema-editor-window ;; name
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 250 400 430)) ;; exterior-box
```



```
) ;; create-visualizations
```

D.2.3 The Royal Road Problem

The royal road problem was the last example presented in Section 7.4. In order to produce this visualization, a new method was created to generate matrices of search space visualizations. This `create-search-space-visualization-matrix` method is presented here along with the code used to produce the example visualization of a GA solving the royal road problem.

```
(defmethod create-search-space-visualization-matrix
  (name-list (dataset ecosystem) chromosome-mapping-technique parent-dialog
    list-of-exterior-boxes coordinate-mapping-technique list-of-list-of-views
    list-of-projection-locus-orderings)

  (mapcar #'(lambda (name exterior-box list-of-views loci-list)
    (create-search-space-visualization
      name
      dataset
      chromosome-mapping-technique
      parent-dialog
      exterior-box
      coordinate-mapping-technique
      list-of-views
      loci-list))
    name-list window-boxes list-of-list-of-views list-of-projection-locus-orderings)
) ;; create-search-space-visualization-matrix
```

This `create-search-space-visualization-matrix` method was applied as follows to produce the visualization shown in Figure 7.14, see page 205.

```

(create-search-space-visualization-matrix
  '(scatterplot-view-0 scatterplot-view-1 scatterplot-view-2
    scatterplot-view-3 scatterplot-view-4 scatterplot-view-5
    scatterplot-view-6 scatterplot-view-7) ;; list-of-names
  run-1 ;; dataset
  'GSM-D-circle ;; chromosome-mapping-technique
  *visualization-dialog* ;; parent-dialog
  '((cg:make-box 400 204 619 402) (cg:make-box 619 204 839 402)
    (cg:make-box 839 204 1058 402) (cg:make-box 1058 204 1278 402)
    (cg:make-box 400 502 619 704) (cg:make-box 619 502 839 704)
    (cg:make-box 839 502 1058 704) (cg:make-box 1058 502 1278 704)) ;; list-of-exterior-boxes
  'D-GSM ;; coordinate-mapping-technique
  (list text-view-0) ;; list-of-views
  '((0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15) (16 17 18 19 20 21 22 23)
    (24 25 26 27 28 29 30 31) (32 33 34 35 36 37 38 39) (40 41 42 43 44 45 46 47)
    (48 49 50 51 52 53 54 55) (56 57 58 59 60 61 62 63))) ;; list-of-projection-locus-orderings

```