Finding diverse examples using genetic algorithms.

Colin G. Johnson.

Computing Laboratory.
University of Kent at Canterbury.
Canterbury, Kent, CT2 7NX, England.
Email: C.G.Johnson@ukc.ac.uk

1 Introduction

The problem of *finding qualitative examples* is an interesting yet little studied machine learning problem. Take a set of objects, $\mathcal{O}$ and a set of classes $\mathcal{C}$, where each object fits into one and only one class. Represent this classification by a total function $f : \mathcal{O} \to \mathcal{C}$. We assume that $|\text{range}(f)| \ll |\mathcal{O}|$.

Sometime this problem is fairly trivial, e.g. is the classification is represented as a database or if there is an easy way to calculate a pseudo-inverse of f. Also in some cases range(f) is a small subset of $\mathcal{C}$, so targeting members of $\mathcal{C}$ is infeasible. One promising approach is to search $\mathcal{O}$.

Clearly in doing this we are assuming that there some kind of underlying structure to f. One approach would be to attempt to uncover this structure in an explicit, symbolic form. However in this paper we use heuristic search methods which search this space without using explicit representations of which areas of the search space are fruitful, et cetera.

There are a number of variants on this problem. In some problem areas we will know in advance how many classes there are, which will give us a stopping criterion, otherwise we will have to use traditional stopping criteria such as GA convergence. In some problem areas the classes will be defined with respect to some metric on $\mathcal{C}$ rather than defined in advance.

2 Motivations.

The main motivation for studying this problem is that it occurs in a wide variety of situations, and that a general heuristic for problems of this type would benefit many different problems (Johnson, 2000). Here are some examples.

Consider creating test data for computer programs. We would like to create one example of a set of data that tests each point in a computer program, for example a set of data that ensures coverage of all lines in a program, or one which ensures that all branches in a program are visited.

A problem with text-based information retrieval systems (Belew, 2000; van Rijsbergen, 1979) is that a single search term can match a number of qualitatively different kinds of object. If we search for a particular person's name, then we might get lots of information about other people with the same name. Careful specification of additional search terms can help, but it can be difficult to find terms or they can over-restrict the search. One idea is to guide the search by looking for pages which are diverse in comparison to currently found documents, and then present the user with a number of distinct examples of documents satisfying their search criteria.

Knot theory is a topic in mathematics which studies the topology of closed loops in space (Adams, 1994; Murasugi, 1996). One common way to study these objects is to investigate the properties of 2D diagrams which are projections of the 3D loops (figure 1). A well-known problem in this subject is finding an example diagram for each of the topologically distinct 3D structures (Hoste, Thistlethwaite, & Weeks, 1998).

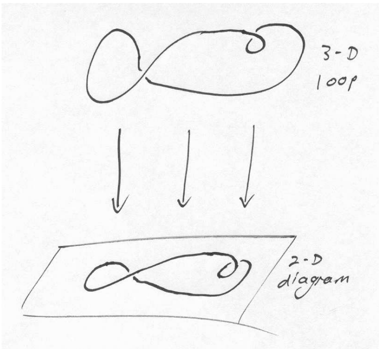


Fig. 1. A knot diagram from a three-dimensional loop.

Another scientific problem occurs in bioinformatics (Attwood & Parry-Smith, 1999). Imagine that we have a large set of rules which tell us how 3D chemical structures of some type are formed from their primary structure. We are interested only in the shape that the molecule presents to the outside world, and we would like to do some experiments in the lab with these chemicals. However there are many possible primary structures for each shape, and we would like to find one of each.

A final application is in the CAD in the broadest sense. Evolution can be used to search a space of designs, including geometrical design, design of networks, architecture and design of sound (Bentley, 1999). One way in which qualitative example finding could be used would be to give the user an overview of the design space by picking out a wide diversity of examples.

3 Review.

In this paper we shall use a genetic algorithm (GA) to search for structured examples. Structured diversity which exploits qualitatively distinct niches in a population is a feature which evolution is good at producing, so GAs are a promising basis for developing qualitative example finding algorithms.

Note that this is different from multimodal optimization using GAs (as surveyed in (Mafoud, 1997)). In multimodal optimization the majority of the effort is in improving solutions within the classes, whereas in our problems moving between the classes is the most important part of the algorithm. There is some common ground with work in AI on computational creativity (Boden, 1990; Partridge & Rowe, 1994), however in our system novelty is a predefined characteristic, not something that needs to be ascertained by the system.

4 Requirements for an algorithm.

In this problem there is no way to extrinsically assign a measure of quality to a solution, i.e. the individual solutions do not have a *fitness*. In order to find solutions to these kinds of problems we might choose to assign a fitness, either by setting intermediate targets or by assigning fitness "on the fly" in each generation, or we might change the selection scheme entirely.

This lack of an external fitness measure means that asking how the algorithms presented below compare with "traditional" GAs is not a meaningful question. GAs are not in themselves optimizers—instead they are robust adaptive systems, variants of which (GA function optimizers—GAFOs) can be used for optimization (De Jong, 1993; Harvey, 1997). This kind of robust adaptivity can be used to apply GAs to problems where the idea of an optimum is absent.

Some basic features of an algorithm for solving such problems:

- Some "substructures" will be capable of being built upon to form objects from many different classes. The algorithm must be able to exploit the fruitful substructures that it "finds", either symbolically or subsymbolically
- The algorithm must be capable of moving quickly onto other areas of search space once an area has been mined out.
- Similarly, the algorithm must recognize when it is in an unproductive area of the search space and move onto other areas.
- The algorithm should not return to unproductive or mined out areas of the search space.

A basic scheme that we can adopt is to use the traditional GA recombination and mutation operators but to create new selection scheme. This is based on an on-the-fly scheme which gives a fitness value to each individual in each generation, based on whether that individual has given evidence that it contains fruitful substructures on which novel solutions can be built.

5 Some test problems.

In this paper we present results from two test problems. In the first problem (the *grid problem*) the solutions consist of strings of letters chosen from the set $\{L, R, U, D\}$. The classification is based around a 100×100 grid of numbers (figure 2), where most of the grid is filled with the number 1, except for four 21×21 regions, each square of which is filled with a different number. For a given solution string we begin by at the point $(50, 50)$ in the grid, and move [L]eft, [R]ight, [U]p or [D]own as we read along the string. The square in which we end up is the class to which that string belongs. If the string ends up outside of the grid it is assigned the class 1.

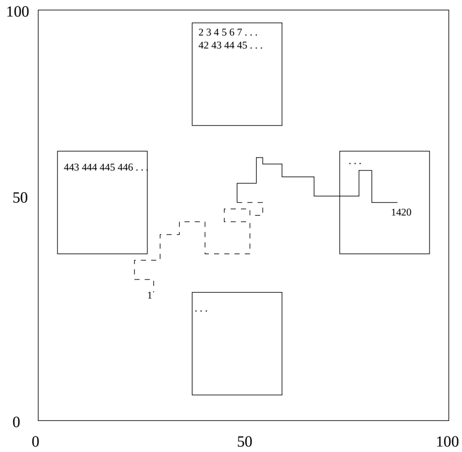


Fig. 2. The grid, with two sample solution paths.

The second problem is a variant on the knot problem described earlier. In this problem we construct knots by *braids*, i.e. sets of strings which run downwards and which cross each other at a finite number of points. We can turn these into knots by joining up the strings with loops (figure 3). Genetic operators are recombination by cutting and rejoining two braids, and mutation by replacing one crossing by a random other crossing at the same position (figure 3). We can associate a polynomial (the *Jones polynomial*) to each braid (Adams, 1994), and search for one example of a braid for each valid polynomial[1].

[1] Thanks to Hugh Morton, University of Liverpool, for supplying computer programs for the calculation of knot invariants.

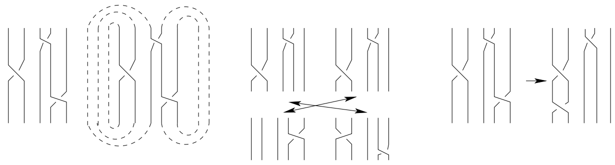


Fig. 3. Operations on braids.

6 Results.

We have been experimenting with a number of variations of genetic algorithms for the test problem. As discussed above there is no extrinsic fitness function, therefore we create a fitness measure on the fly in each generation.

6.1 On the fly fitness allocation.

The simplest way to do this is to give a score of 1 to those members of the current population which are "novel", defined to mean that their classifications have not occurred in any previous generations, and 0 otherwise. We have also investigated a variation on this, which attempts to strengthen those individuals which have fruitful substructures within them by assigning a higher score to those solutions which come from parent solutions that have produced a diverse range of solutions. Firstly we create a new population by crossover and mutation, using the fitness values generated in the previous generation. We then tally up the total number of novel children had by each parent, and then assign this total score to each of their children (figure 4). This is then used as a fitness measure in a traditional GA framework. Comparisons are given in figure 5(c).

6.2 Crossover with random strings.

In some experiments we introduced a number of random strings into the population at each generation. The aim of this was to see whether these strings would crossover with other strings which contain good substructures but help to exploit those substructures. This was tried for a number of problems and proved unsuccessful—see figure 5(a) for the results for the grid problem.

6.3 Mutation rates.

Experiments have been carried out with a number of different mutation rates (figure 5(b)). Over a number of experiments, including the one outlined here, the most successful mutation rate was typically much higher than the mutation rate for GAs used for optimization, with a mutation rate of 0.1 being

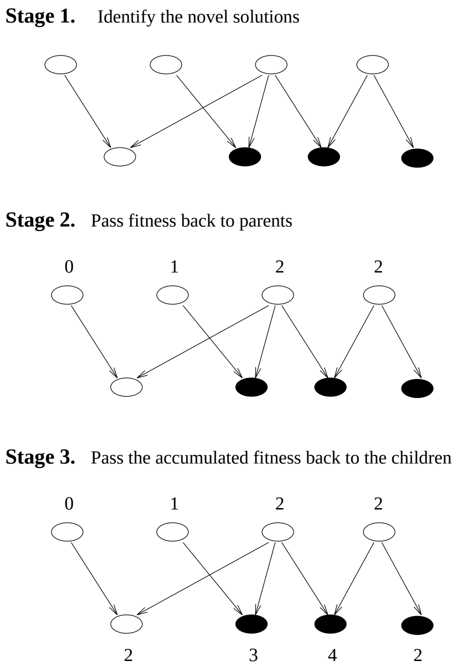
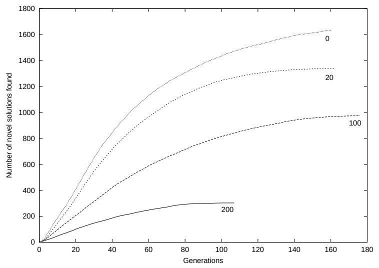


Fig. 4. An algorithm for enhancing fruitful substructures. Those structures which come from parents which have the most number of novel children receive the highest fitness.

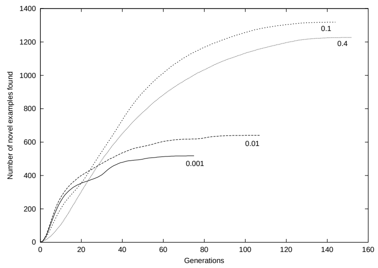
best contrasted with rates of 0.01–0.001 (Mitchell, 1996) or the reciprocal of bitstring-length (Bäck, 1996) for GAFOs. This is likely to be caused by the mutation being a major source of the ability of this algorithm to find novel solutions based around existing ones, rather than being a "background operator" to prevent convergence as with GAFOs.

6.4 Comparisons with random search.

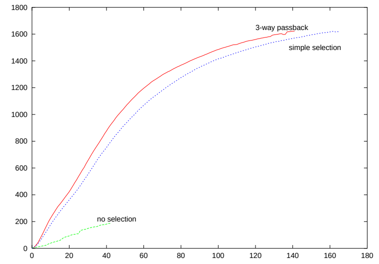
Figures 5(d) and 5(e) present comparisons against random search for these two test problems. One difficulty here is that random search gives a good performance anyway for the grid problem, so a major next step in this kind of work is finding better test problems which are harder to solve by brute force methods. There is a more general difficulty in presenting results on problems such as these. An aim of the research is to demonstrate that general heuristics can be used for these kinds of problems, but criteria for success are problem dependent, but for test problems we have no criteria for success and comparison with random search is not ideal.



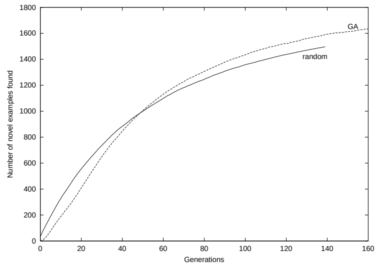
(a) Random strings (grid)



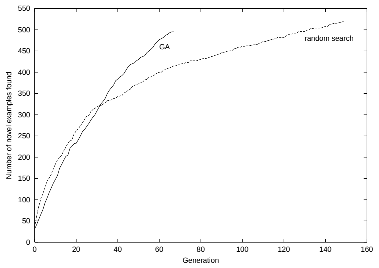
(b) Mutation rates (grid)



(c) Selection schemes (grid)



(d) Comparison with random (grid)



(e) Comparison with random (braid)

Fig. 5. Results.

7 Future work.

Other variants on the algorithm are worthy of further investigation, e.g. preliminary experiments on using hypermutation to prevent the algorithm getting stuck when the number of novel solutions is small have proven fruitful. Future work will include looking for rigourous, tunable test problems, e.g. analogies with royal road functions (Mitchell, Forrest, & Holland, 1992), and deceptive problems (Goldberg, 1987); theoretical analyses, e.g. investigating how Holland's schema theorem (Holland, 1975) might explain the discovery and exploitation of fecund substructures; and application to real-world problems.

References

Adams, C. A. (1994). *The knot book.* W.H. Freeman.

Attwood, T., & Parry-Smith, D. (1999). *Introduction to bioinformatics.* Addison Wesley Longman.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice.* Oxford University Press.

Belew, R. K. (2000). *Finding out about : information retrieval and other technologies for seeking knowledge.* Cambridge University Press. (In preparation)

Bentley, P. J. (Ed.). (1999). *Evolutionary design by computers.* Academic Press.

Boden, M. (1990). *The creative mind: Myths and mechanisms.* Abacus.

De Jong, K. (1993). Genetic algorithms are NOT function optimizers. In L. Whitley (Ed.), *Foundations of genetic algorithms 2* (pp. 5–17). Morgan Kauffmann.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In L. D. Davis (Ed.), *Genetic algorithms and simulated annealing.* Morgan Kaufmann.

Harvey, I. (1997). Cognition is not computation: Evolution is not optimisation. In W. Gerstner, A. Germond, M. Hasler, , & J.-D. Nicoud (Eds.), *Proceedings of the seventh international conference on artificial neural networks* (pp. 685–690). Springer-Verlag.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* MIT Press. (Second edition 1992)

Hoste, J., Thistlethwaite, M., & Weeks, J. (1998). The first 1,701,936 knots. *The Mathematical Intelligencer*, *20*(4), 33-48.

Johnson, C. G. (2000). Understanding complex systems through examples: a framework for qualitative example finding. In P. A. Gelepithis (Ed.), *Complex intelligent systems.* Kingston University.

Mafoud, S. W. (1997). Niching methods. In T. Bäck, D. B. Fogel, & Z. Michalewicz (Eds.), *Handbook of evolutionary computation* (pp. C6.1.1–C6.1.4). Oxford University Press / Institute of Physics.

Mitchell, M. (1996). *An introduction to genetic algorithms.* Bradford Books/MIT Press.

Mitchell, M., Forrest, S., & Holland, J. (1992). The royal road for genetic algorithms : Fitness landscapes and GA performance. In F. Varela & P. Bourgine (Eds.), *Towards a practice of autonomous systems : Proceedings of the first european conference on artificial life.* MIT Press.

Murasugi, K. (1996). *Knot theory and its applications.* Birkhäuser.

Partridge, D., & Rowe, J. (1994). *Computers and creativity.* Intellect Books.

van Rijsbergen, C. J. (1979). *Information retrieval.* London: Butterworths.