Encoding Scheme Issues For Open-Ended Artificial Evolution

Nick Jakobi

School of Cognitive and Computing Sciences
University of Sussex, Brighton BN1 9QH, England
email: nickja@cogs.susx.ac.uk
tel: (UK) 01273 678061
fax: (UK) 01273 671320

Abstract. This paper examines the ways in which the encoding scheme that governs how phenotypes develop from genotypes may be used to improve the performance of open-ended artificial evolution for design. If an open-ended framework involving variable complexity genetic algorithms is adopted, then the vast majority of the evolutionary effort is spent exploring neutral flat areas of the search space. Domain-specific heuristics may be employed to reduce the time spent on searching these neutral areas, however, and the ways in which domain knowledge may be incorporated into the encoding scheme are examined. Experiments are reported in which different categories of scheme were tested against each other, and conclusions are offered as to the most promising type of encoding scheme for a viable open-ended artificial evolution.

1 Introduction

Genetic Algorithms (GAs) are primarily employed as optimization techniques on problems which involve finding satisfactory values for a given, fixed number of parameters. However, GAs are one of the few techniques that may also be used on design problems, where the size of the optimal parameter set, and the complexity of satisfactory solutions, is not known beforehand. In these situations, the size (and sometimes structure) of the genotype is put under evolutionary control resulting in what shall be referred to throughout this paper as variable complexity GAs [4, 3]. Areas in which these have been tried include aircraft wing design [2], computer programming [6], and the design of control architectures for autonomous agents [5]. Although some success has been reported in all of these fields, an artificial evolution that is a general alternative to human design ingenuity has not yet been achieved. This paper looks at the form that such an open-ended evolutionary process would have to possess, and compares and contrasts ways in which domain-specific heuristics may be applied using the encoding scheme to ensure that the time periods involved remain within sensible limits.

2 Open-ended artificial evolution

There is nothing magic about an evolutionary process capable of producing better designs than a human. What is hard, is to produce better designs than a human within a reasonable period of time. This section examines the basic form of open-ended evolutionary process we will be dealing with in this paper, and section 3 and 4 look at how the performance of this process may be improved.

The term 'open-ended' is here used to refer to a necessary feature of a viable evolutionary design alternative, namely that the process will continue to produce better and better solutions as long as there are better solutions to produce. Thus, in an open-ended evolutionary scenario, if greater or lesser complexity is *required* for fitter solutions, then it *will* evolve. Eventually and inevitably, such a process will evolve better designs than a human designer.

Although at first sight it may seem miraculous, there are in fact many simple methods by which the open-endedness of the evolutionary process may be ensured [1]. The one considered in this paper, however, seems the most amenable to performance improvements [8]. In its simplest form, restrictions on the way in which genotypes code for phenotypes ensures that the search space upon which a variable-complexity GA operates contains no *local* fitness maxima. Since there are no local fitness maxima in the search space, the GA will only come to rest on a global maxima if there is one, or not at all if there isn't one (in a coevolutionary situation for example). To understand how this is possible, a brief analysis of the nature of variable complexity search spaces is given below, and then a simple example encoding scheme is put forward that ensures an open-ended evolutionary process.

The way in which genotypes are represented and the nature of the genetic operators determines the (potentially infinite) set of possible genotypes upon which a particular GA operates. If a GA is employed to optimize a fixed number n parameters, then the set of possible genotypes can be, and usually is, visualized for explanatory purposes as an n dimensional Euclidean space. This is normally referred to in the optimization literature as 'genotype space'. The set of possible genotypes searched by a variable complexity GA, however, is best thought of for the purposes of this paper as a graph or network (i.e. a set of interconnected nodes) rather than a fixed dimensionality space. Each node on such a graph corresponds to a possible genotype and each connection between nodes corresponds to a single application of a genetic operator.

In order to make this more explicit, consider as an example a stripped down version of a variable complexity GA acting upon genotypes of finite cardinality. It involves two genetic operators: mutation, as normally conceived, and a 'change size' operator which adds and/or subtracts material to and/or from the genotype, thus changing the size of the genotype and the complexity of the solution it encodes. Two genotypes, under such a GA, are maximally similar, though not identical, if they differ by a single genetic base unit (bit, character, nucleic acid

[1] The most trivial of these involves the creation of a totally random genotype at regular time intervals.

etc.). This is the case if *either* they are the same size and shape but differ by a single genetic base unit (the mutation operator) *or* one genotype contains exactly one more genetic base unit than the other but in all other respects they are identical (the 'change size' operator). A connection is defined to exist between two nodes on the graph of possible genotypes if and only if their corresponding genotypes are maximally similar. The more similar two genotypes are to each other, the shorter the direct path between their corresponding nodes.

A variable-complexity GA, then, is regarded here as searching a graph of possible genotypes, rather than a Euclidean genotype space, for nodes that correspond to phenotypes of high fitness. A local fitness maximum on such a graph is constituted either by a single node or by a subnetwork of inter-connected nodes that are all of the same corresponding fitness, and may be recognised by the fact that *all* the nodes directly connected to it are of lower corresponding fitness. If there are no local fitness maxima on the graph of possible genotypes then the evolutionary process may be considered open-ended. Encoding schemes that ensure this may take many forms and, in fact, open-endedness of this nature turns out to be trivial. An example encoding scheme is outlined below.

A simple encoding scheme that precludes local fitness maxima

Consider a scheme in which it is always possible to add extra genetic material (in the form of extra bits, characters etc) to the genotype without effecting the phenotype, and it is always possible to switch segments of the genotype 'on' or 'off' by way of single point mutations. These restrictions may seem strange but they are in fact true of the encoding scheme behind natural development. Now consider a worst case scenario - the genotype coding for a particular phenotype cannot undergo any normal single point mutation anywhere without suffering a loss in fitness. Extra genetic material that is 'off' can always be added to the genotype without effecting the phenotype, however, and this will eventually lead, after a monkeys-typing-Shakespeare length of time, to the evolution of a stretch of 'off' genotype that codes for a fitter phenotype (if there is one) than that expressed by the current 'on' stretch of genotype. Since a single-point mutation can always switch an 'on' stretch of genotype to 'off' and an 'off' stretch of genotype to 'on', we may expect that *eventually* 'junk DNA' that produces a fitter phenotype is switched 'on' while the rest of the genome is switched 'off'.

Under this or any number of more subtle domain-specific encoding schemes that display similar properties, we can guarantee that no node or set of nodes on the graph of possible genotypes constitutes a local fitness maximum. All nodes will connect to at least a few other nodes of the same corresponding fitness thus forming large *neutral networks* (see [9]). In every neutral network there will be one or more nodes that also has a connection to a node in a neutral network of higher fitness (if there is one). It is always possible, therefore, to find a path from *any* node through the graph of possible genotypes that monotonically increases[2], with respect to corresponding fitness, ad infinitum.

[2] monotonically increasing means never going down, not always going up.

3 Applying domain-specific heuristics to neutral network search

What is immediately obvious, if one excepts the basic open-ended framework laid out above, is that by far the largest proportion of the computational effort will be spent on blindly searching neutral networks as opposed to climbing hills. This is not only because hill-climbing type search *is* directed, but also because nodes on the graph of possible genotypes that are directly connected to nodes of greater fitness will be in the vast minority. Thus although other techniques (such as simulated annealing) may out-perform artificial evolution at hill-climbing type search, there is a certain amount of evidence [1, 7] that artificial evolution is better suited to neutral network search than these other techniques, and should therefore constitute the search method of choice on the sort of open-ended design problems this paper is concerned with. Neutral network search is, however, non-directed by definition, and if open-ended artificial evolution is to be a viable alternative to human design then all attention must be focused on ways to speed up the efficiency of neutral network search from the unacceptable monkeys-typing-Shakespeare levels reported above. This section looks at what it would mean to apply domain-specific heuristics to this search and general ways in which this can be accomplished using the encoding scheme.

As an example of what it means to apply a heuristic to open-ended artificial evolution, we shall consider a certain property p of phenotypes in a particular design domain such that *in general* successful phenotypes display p rather than $p\prime$ (i.e. not p). The property p could be symmetry, for example, or large amounts of repeated structure, or even the existence of specific configurations of elements within the phenotype. In addition, we notice that the instantiation of p in a *random* phenotype is very unlikely and are thus lead to suspect that the success of phenotypes that display p is somehow causally related to p. By concentrating the evolutionary search on genotypes that code for p (i.e. which correspond to phenotypes that display p), therefore, we are employing the hypothesis that p contributes to phenotypic success as a heuristic. If the heuristic is well-founded then we may expect to decrease the time taken to evolve satisfactory results.

There are two ways in which the encoding scheme can be employed to achieve this (both of which are explained below) by increasing the relative frequency of nodes that code for p throughout the *entire* graph of possible genotypes. First though, it is important to understand exactly why increasing the relative frequency of p genotypes should improve the performance of neutral network search. The heuristic implies that for any particular neutral network, the nearest area of higher fitness is more likely to be constituted by nodes that code for p than nodes that code for $p\prime$. This is true both in the case where the neutral network is itself constituted wholly by nodes that code for p, and in the case where it is constituted by a mixture of nodes that code for p and nodes that code for $p\prime$. In both of these two cases, increasing the relative frequency of p nodes throughout the entire graph of possible genotypes will increase the proportion of new offspring that are p, thus increasing the rate at which p genotypes are sampled and decreasing the time taken to find the area of higher fitness. Fur-

thermore, in the latter case, increasing the relative frequency of p nodes will also increase the proportion of individuals in the population that code for p rather than $p\prime$, thus increasing the likelihood of p genotypes being picked as parents in offspring events, and thus again increasing the proportion of new offspring that are p. Therefore, whether at the beginning of the evolutionary process with a mixture of p and $p\prime$ individuals in the population, or at an advanced stage where only primarily p individuals remain, an increased relative frequency of p nodes throughout the graph of possible genotypes will continue to do work in improving the performance of neutral network search.

A simple *direct* encoding scheme (where phenotype space is mapped evenly onto the graph of possible genotypes) may be altered to increase the relative frequency of nodes that code for a certain property p in only one of two distinct ways. Either the encoding scheme is *biased* so that there are lots of ways in which genotypes may code for p but only a few ways in which a genotype may code for $p\prime$, or it is *restricted* so that genotypes are incapable of coding for some or all of the phenotypes that display $p\prime$. The first method uniformly increases the proportion of nodes on the graph of possible genotypes that code for p, and the second method uniformly decreases the number of nodes on the graph of possible genotypes that code for $p\prime$. In what follows, the encoding schemes that fall into each of these two categories shall be referred to as biased and restricted encoding schemes respectively. There is of course a third category that involves both biasing the search in favour of phenotypes that display p and restricting the ways in which phenotypes may display $p\prime$ and this will be referred to as a *hybrid* encoding scheme.

4 Testing the different categories of encoding scheme against each other

This section reports the results of experiments designed to compare the relative abilities of the four different categories of encoding. Two different sets of experiments were performed, the first designed to see just what sort of performance improvements we can expect from applying heuristics to neutral network search, and the second designed to see what happens if the heuristic that is applied is not well-founded. In the first set of experiments specifically devised direct, restricted, biased and hybrid encoding schemes were tested against each other on their ability to apply a heuristic, namely that fit phenotypes were symmetrical, to a certain symmetrical evolutionary task. In the second set of experiments only the biased and direct encoding schemes were tested against each other, but this time on an asymmetrical evolutionary task. In both experiments, schemes were judged on the average time taken by a GA to search a neutral network of a particular fitness for nodes in a second fitter network. Precise details of the neutral networks and encoding schemes are given below.

Phenotypes consisted of the pattern of filled squares on an eight by eight grid. Their fitnesses were assigned according to the templates shown in figure 2. As can be seen from the diagrams, in both experiments the templates of fitness

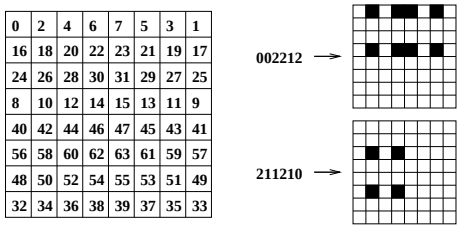

Fig. 1. The numbering system used to fill in squares on the phenotypic grid. On the right are two example genotype fields from the biased encoding scheme, and their corresponding phenotypic effect

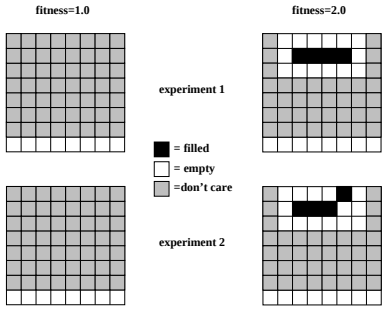

Fig. 2. This figure shows the phenotypic templates that were used to define neutral networks on the graph of possible genotypes.

2.0 are special cases of the templates of fitness 1.0. This means that the two neutral networks formed by nodes which correspond to phenotypes that match the templates of fitness 2.0 are both sub-networks of the huge neutral network formed by the nodes which correspond to phenotypes that match the templates of fitness 1.0. In experiment 1, the template of fitness 2.0 is symmetrical and in experiment 2, the corresponding template is not.

The genotypes in the direct encoding scheme consisted of a precise number n of 6-bit binary fields joined together to make a string, where n was under genetic control. In order to decode a phenotype from a genotype, a decimal value is calculated from the binary value of each field, and the relevant square filled in on the grid according to the special numbering shown in figure 1. Thus each phenotype has as many squares filled in as there are fields on the genotype.

The genotypes in the biased encoding scheme consisted of a precise number n of 6-bit *ternary* fields, where n was under genetic control. Of the three possible bit values, 0 represents 0, 1 represents 1, but 2 is a wild card character. Therefore each 6-bit ternary field maps to 2^k 6-bit binary numbers, where k is the number of 2s in the ternary field. When decoding a ternary field, a square is filled in on the grid (according to the special numbering shown in figure 1) for every binary number that matches. Two examples of decoded fields are given in figure 1. It is important to note that although the biased encoding scheme is capable of encoding any phenotype, genotypes that code for a high degree of symmetry and order will be much more common than those that don't.

The genotypes in the restricted encoding scheme consisted of a precise number n of 5-bit binary fields, where n was under genetic control. During decoding, each field is treated as if it had an extra bit of value 2 on the end, and decoded in the same way as for the biased encoding scheme. This means that for each field on the genotype a symmetrical pair of squares is filled in on the phenotype.

All phenotypes are vertically symmetrical.

The genotypes in the hybrid encoding scheme consisted of a precise number n of 5-bit ternary fields, where n was under genetic control. During decoding, each field is treated as if it had an extra bit of value 2 on the end and decoded in exactly the same way as if it were a six-bit ternary field of the biased encoding scheme. This means that many more squares may be filled in on the phenotype than there are fields on the genotype and that all phenotypes are vertically symmetrical.

4.1 Experimental results

In both experiments, the GA[3] was run 500 times for each encoding scheme. On every run the population was started from the same point on the first neutral network, namely the node that corresponds to a genotype of length 0 (which represents a blank grid under every encoding scheme). On each run, the number of fitness evaluations performed before the GA found the second neutral network was counted and assigned as the score for that run.

A table of the average scores per five hundred runs per encoding scheme per experiment is given in Table 1. Because the distributions underlying the numbers of evaluations taken approximate a Poisson distribution, the variances of the scores are of the same order of magnitude as the means, and conventional graphical representations are largely uninformative. To produce Figures 3 and 4, the scores from each set of five hundred runs were sorted in order of magnitude and plots were made of rank against value (i.e. the number of evaluations taken to find the neutral networks of fitness 2.0 on each run). This give a fairly good picture of the abilities of each encoding scheme to apply heuristics to neutral network search. The shallower the gradient, the better.

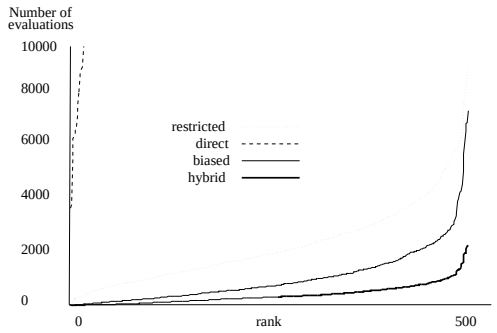

Fig. 3. Results of experiment 1 (see text).

The first thing to say about these results is that they prove quite conclusively that the choice of an appropriate encoding scheme which exploits domain knowledge in a heuristic way, does indeed significantly speed up neutral network search. Of the four encoding schemes tested in experiment 1 only the direct encoding scheme does *not* employ symmetry as a heuristic and this makes it two orders of magnitude worse than its nearest rival. The hybrid encoding scheme, which employs the heuristic in a dual way, seemed to work the best.

In experiment 2, an asymmetrical template defines the neutral network of fitness 2. Only the biased and the direct encoding schemes are capable of rep-

[3] Population 100. Steady state. Tournament selection. Genetic operators included a mutation operator and a 'change size' operator capable of adding or subtracting a random genotype field. These operators were applied at a rate of *one* (of the two operators) applications per offspring event in the of ratio 4:1, respectively. A modified form of single-point crossover was employed at all offspring events.

encoding scheme	*experiment 1*	*experiment 2*
direct	175220	133700
biased	1054.1	1650600
restricted	2312.4	
hybrid	404.5	

Table 1. The average number of evaluations for each encoding scheme, taken over a series of 500 runs, that the GA took to find the neutral network of fitness 2.

resenting a phenotype of this type, and the experiment was performed in order to test the biased scheme's ability to search for asymmetrical phenotypes when it is biased in favour of symmetrical ones. As can be seen, the biased encoding scheme performs an order of magnitude *worse* than the direct encoding scheme on this task. This is not surprising if one considers that increasing the relative frequency of nodes that code for symmetry on the graph of possible genotypes decreases the relative frequency of nodes that code for asymmetry.

4.2 A hybrid restricted scheme is advocated

No definite conclusions can be drawn from these results as to which type of encoding scheme is the most promising for a viable open-ended evolutionary design alternative. Filling in squares on a grid is not the same as evolving complex designs for real world problems. The results are, however, extremely suggestive. Even though the performance difference between the hybrid encoding scheme and the biased encoding scheme seem slight, it is nevertheless some sort of (biased) restricted encoding scheme rather than a pure biased encoding scheme that is advocated here.

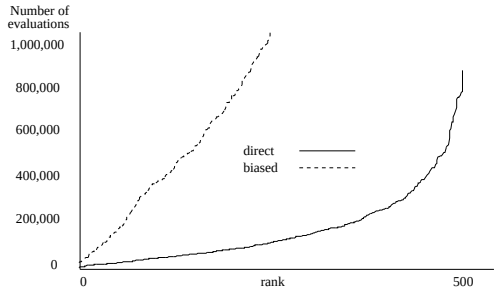

Fig. 4. Results of experiment 2 (see text).

To appreciate why, consider how each encoding scheme could apply heuristics to the task of evolving the design for something that displayed a great deal of repeated structure such as a millipede, for example. A hybrid restricted scheme could encode the design for a single leg once on a genotype and reuse this code 1000 times. Perhaps it would also code for a small number of position dependent operators, such that the front legs differed from the back legs according to some spatial transformation (see [10] for examples of this sort of thing occurring in nature). A true biased encoding scheme, on the other hand, would have to safeguard the possibility that any leg may evolve independently to any other, since every phenotype is potentially representable. Thus, although it could encode the design

for a millipede in a similar way to the hybrid restricted scheme (by repeating the code for 1 leg 1000 times), it would also have to be able, at least in principle, to code for each of the 1000 legs individually. However it is done, this will involve a longer or higher cardinality genotype resulting in a larger search space with a lower relative frequency of nodes that code for the sort of repeated structure we are after.

Many researchers will feel uneasy about the idea of adopting encoding schemes for open-ended artificial evolution that are unable to encode all possible phenotypes. They may feel that, even if this does lead to performance increases, it amounts to pre-judging the problem and prevents the evolution of unexpected solutions - precisely the reason evolution is being used in the first place. There is one further thing to be said, however, that may quell these fears. An encoding scheme that is heuristically biased for the evolution of certain traits, is biased *against* the evolution of other traits. This is clear from experiment 2. Therefore in order for a biased scheme to even come close to competing with a hybrid restricted scheme at evolving complex phenotypes (such as in the millipede example above), the level of bias would be such that the evolution of a particular phenotype which the scheme is biased *against* would take a totally impractical length of time. In fact, as can be seen from experiment 2, it would actually take many, many orders of magnitude longer to evolve than if a simple direct encoding scheme was used. Thus, by removing the possibility of representing such phenotypes, we are actually losing very little.

5 Conclusions

The open-ended artificial evolution of designs that are better than a human designer could produce *is*, at least as far as the evolutionary process is concerned, possible. In fact, as was shown in section 2, it is trivial..... given enough time. What also became clear in this section was that directed hill-climbing type search plays a minor role in this sort of open-ended evolutionary process, and it is the non-directed search of neutral areas of the space that constitutes the vast majority of the evolutionary effort.

Section 3 explained how domain-specific heuristics could be applied to decrease the time spent by the evolutionary process on searching neutral networks. This is a pressing requirement if open-ended artificial evolution is to be a viable alternative to human design. The four different categories of encoding scheme were introduced, and broad explanations were given as to how each category incorporates domain knowledge to speed up the process.

Finally, in section 4, specially devised instances of the different categories of encoding scheme were tested against each other. Although the three types of encoding scheme that employed heuristics out-performed a simple direct encoding scheme (by several orders of magnitude) on a task where the heuristics were well-founded, it was shown that the direct encoding scheme was in fact the best performer on a task where the heuristics were *not* well-founded.

Since the time taken by a direct encoding scheme to evolve anything really complicated is assumed to be totally impractical, it seems that the only scenario in which open-ended artificial evolution can ever be a viable alternative to human design is if domain-specific heuristics are employed *and* these heuristics are well-founded. There is no point in making allowances for the scenario in which the heuristics are not well-founded, since if this is the case then evolution will take even longer than a direct encoding scheme to evolve anything complex or useful. The type of encoding scheme that is advocated for open-ended artificial evolution, therefore, is one that heuristically restricts the evolutionary search space.

Acknowledgements

I would like to thank Phil Husbands, Inman Harvey and others at the School of Cognitive and Computer Sciences for various crucial discussions. Thanks also to the school of COGS itself for the bursary that allows me to undertake this work.

References

1. M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *Journal of Physical Chemistry*, 92:6881–6891, 1988.
2. P.J. Gage and I.M. Kroo. A role for genetic algorithms in a preliminary design environment. *A.I.A.A.*, 1993.
3. P.J. Gage, I.M. Kroo, and I.P. Sobieski. A variable-complexity genetic algorithm for topological design. *A.I.A.A.*, 33(11), 1995.
4. I. Harvey. Species adaptation genetic algorithms: the basis for a continuing saga. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354, Cambridge, Massachusetts, 1992. M.I.T. Press / Bradford Books.
5. I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994.
6. J. Koza. *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, Massachusetts, 1992.
7. Phil Husbands Malcolm McIlhagga and Robert Ives. A comparison of search techniques on a wing-box optimisation problem. In *Proceedings of Parallel Problem Solving in Nature*, Berlin, Germany, 1996. Springer-Verlag.
8. N.Jakobi. Facing the facts: Necessary requirements for the artificial evolution of *complex* behaviour. Cognitive Science Research Paper CSRP422, University of Sussex, 1996.
9. P. Schuster. Artificial life and molecular evolutionary biology. In *Proceedings of the European Conference on Artificial Life*, pages 3–19. Springer-Verlag, 1995.
10. D.W. Thompson. *On Growth and Form*. Cambride University Press, 1942.