USING GENETIC ALGORITHMS WITH SMALL POPULATIONS *

Colin R Reeves
School of Mathematical and Information Sciences
Coventry University
UK
Email: CRReeves@coventry.ac.uk

Abstract

Most reported (serial) implementations of genetic algorithms have assumed population sizes of at least 30, and often very much larger. The general question of population sizing has been considered from several aspects, with somewhat conflicting conclusions, and then only for populations of binary strings. In this paper we consider applications where it is important to use as small a population as possible, where the number of fitness evaluations is limited, and where non-binary representations are important.

First, we approach the question of specifying a *minimum* size by asking what characteristics of an initial population are likely to lead to poor performance, and calculate population sizes which will almost certainly avoid such features, on the assumption that initial populations are chosen in a random fashion. It will be shown that rather small populations suffice for binary strings, but that populations need to be considerably larger for the more general case where string alleles are defined over q-ary alphabets. This also gives an interesting sidelight on the debate in respect of the general desirability of using higher-cardinality alphabets.

We then consider whether we should choose initial populations other than randomly, and make some connections with work in the fields of *error-detecting codes* and *experimental design*. Finally, we report some computational comparisons which suggest that using a structured initial population may be helpful.

*Published in S.Forrest (Ed.) (1993) *Proceedings of the* 5th *International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA.

1 INTRODUCTION

One of the most obvious questions relating to GA performance is how it is influenced by population size. In principle, it is clear that small populations run the risk of seriously under-covering the solution space, and so increase the chance of premature convergence to a poor solution. On the other hand, larger populations allow the exploration of fewer generations per unit of computational effort, and if the available computational effort is limited, may preclude convergence at all. Some research has been reported on this problem in [Goldberg, 1985]. Using the criterion that we should try to maximise the expected number of new schemata per individual, this work indicates that the optimal size for binary-coded strings grows exponentially with the length of the string n. Some refinements of this work are reported in [Goldberg, 1989a], but they do not change the overall conclusions significantly.

However, this would imply extremely large populations in most real-world problems, and the *practical* performance of the GA would be quite uncompetitive with other optimization methods such as simulated annealing and tabu search. Fortunately, empirical results from many authors (see e.g. [Grefenstette, 1986] or [Schaffer *et al.*, 1989]) suggest that population sizes as small as 30 are quite adequate in many cases, while some experimental work [Alander, 1992] suggests that a value between n and $2n$ (where n is the string length) is optimal for some problem types. Some of Goldberg's later work, based on a different argument [Goldberg and Rudnick, 1991], goes some way to supporting the use of populations rather smaller than his earlier work suggested. Nevertheless, little has been published that is relevant to *really* small populations.

1.1 SMALL POPULATION APPLICATIONS

This research was motivated by an attempt to apply genetic search methods to problems of engineering design where the effect of using a given set of parameters has to be determined by experiment. (In some cases the problem may arise in a slightly different way, where the evaluation of the engineering design requires a lengthy computer calculation, as in [Keane, 1993] for example.) Such experiments may be costly to set up and run, and thus the number of experiments needs to be kept as low as possible. Current practice is normally to use standard methods of *experimental design*, or modern derivatives such as those associated with the names of Taguchi [Taguchi, 1986] or Shainin [Shainin and Shainin, 1988]. Such methods necessarily make some drastic (and often unfounded) simplifying assumptions about the nature of the underlying process—for instance, that the parameter *effects* are additive, and that no *interactions* occur.

Given that GAs have frequently been shown to perform well on non-linear problems, even in the presence of noise, it seemed worth the attempt to apply genetic search for a good set of parameter values in such problems. However, there is an obvious problem in the limitation on the number of experiments, which rules out the use even of moderately-sized populations in many cases. For example, in the chemical industry, it is not uncommon to have a problem with 9 parameters (or *factors*), each of which could take 3 values (or *levels*). In this situation, a typical experimental design might call for 27 experiments. These would almost certainly be replicated a few times, but it would be unlikely that the total number of experiments would much exceed 100. A typical GA with an initial population of 30 experiments would only just be getting into its stride, unless the underlying process were well-behaved (i.e. linear), in which case the experimental design approach should win hands down anyway. Using what we have elsewhere [Reeves, 1993] called an *incremental* replacement strategy—what others (see e.g. [Syswerda, 1989]) have called a *steady-state* GA—would also be preferable to the more traditional generational GA using *block* replacement, but this still does not answer the question of population size.

To use a GA effectively in applications such as those described, we would therefore inevitably need to start with smaller populations—but how much smaller? It seemed essential that we first try to characterise a *minimal* population size, below which the GA could not be expected to operate effectively.

2 MINIMUM POPULATION SIZES FOR q-ARY ALPHABETS

The initial principle that we adopted is that, at the very least, every possible point in the search space should be *reachable* from the initial population by crossover only[1]. It is clear that this requirement can only be satisfied if there is at least one instance of every allele at each locus in the whole population of strings.

It is almost universally assumed that the initial population consists of strings chosen using a discrete uniform distribution over the appropriate alphabet at each locus. On this assumption we can calculate the probability P_q^* (for a q-ary alphabet) that at least one allele is present at each locus in the initial population. For binary strings, this is very straightforward: the probability that an allele (0 or 1) is missing at a particular locus in a population of M strings is clearly given by

$$(1/2)^{M-1}.$$

Hence, for strings of length L,

$$P_2^* = (1 - (1/2)^{M-1})^L.$$

For q-ary alphabets, the calculation is somewhat less straightforward. What we need to compute (for each locus) is the number of possible assignments of q symbols to M positions such that every symbol is used at least once. This may be related to a standard problem in combinatorics, whence it can be shown that this number is given by

$$q!S(M,q)$$

where $S(M,q)$ is the *Stirling number of the second kind*, which can most simply be generated by the recursion

$$S(M+1,q) = S(M,q-1) + qS(M,q); \; M \geq 1, \; q \geq 2;$$

where clearly

$$S(M,1) = 1 \; \forall M.$$

(It is also necessary to define $S(1,2) = 1$ in order to apply the formula. For further details see a standard text on combinatorics such as Liu [Liu, 1968]).

Since the total number of assignments without restriction is q^M, we find that

$$P_q^* = \left\{ \frac{q!S(M,q)}{q^M} \right\}^L.$$

It is now possible to determine values of M such that the probability P_q^* exceeds a value α. Some examples are given in the curves of Figures 1-3 for the cases $\alpha = 95\%$, 99% and 99.9%.

For the binary case, it can be seen that the minimum population size is relatively small, even for strings

[1]By including mutation, it is of course possible to reach every point regardless of the initial population. An alternative might thus be to compensate a small population with a high mutation rate. This trade-off has been remarked on before in, for example, [Schaffer *et al.*, 1989] in analysing interactions between different GA parameter settings. However, the power of GAs is usually thought to be associated with the recombinative effect of crossover, so we ignored this possibility, at least initially.

much longer than those in which we are interested in this work. However, this does help to account for the good reported performance of GAs with populations as low as 30 for a variety of problems. It can be seen that in a randomly generated population of binary strings we are virtually certain to have every allele present for populations of this size.

For q-ary strings ($q > 2$), however, the situation is not so promising for the applications we have in mind: even for short strings of say 10 bits, we need a population of at least 20 strings to be almost certain of covering every allele in the case $q = 3$. For larger alphabets, minimal population sizes are substantial even for short strings.

3 THE BINARY-CODING DEBATE

This analysis also casts an interesting sidelight on the debate in respect of alphabet sizes. Holland [Holland, 1975], and following him Goldberg [Goldberg, 1989b], stressed the advantage of a binary alphabet, in that it allows the sampling of the maximum number of schemata per individual in the population. More recently, Antonisse has put forward a counter-argument in [Antonisse, 1989] by re-defining the concept of a schema. In traditional (binary) GAs, a schema is a string defined on the characters $\{0, 1, *\}$, where the $*$ symbol represents a 'don't care' condition: i.e. a string containing at least one $*$ represents a *set* of strings in which the $*$ may be replaced by either a 0 or a 1. Antonisse argues for interpreting the $*$ as representing any *subset* of the available symbols, with the implication that the higher-cardinality alphabet actually samples *more* schemata.

In practice, however, the sampling of schemata is only possible if the individuals in the population can actually represent them. What this analysis shows is that we need proportionally larger populations if this criterion is to be met for higher-cardinality alphabets. We can compare, for example the cases $q = 2$ with $q = 8 = 2^3$. In the second case, we would have a string 3 times as long if we were to use a binary encoding of the 8 symbols in the larger alphabet. However, the minimal population size for an 8-ary string of length L is about 6 times that needed for a binary string of length $3L$. Similarly, for $q = 16$, we find that the minimal population size is rather more than 9 times larger than an equivalent binary string of length $4L$. Thus, for a given problem, the computational burden is likely to be much greater for the higher-cardinality alphabet to achieve the same results, even allowing for the necessity of converting the binary code into the appropriate value in the q-ary alphabet.

This is intuitively reasonable, of course. To represent all the 8 possible alleles of an 8-ary alphabet at one locus we clearly need a minimum of 8 strings, whereas it is possible to generate all 8 by different combinations of just 2 binary strings. Thus the normal genetic operators can still generate the alleles that are not explicitly present in an initial binary population. We also suspect (although this has yet to be tested) that a GA using a higher-cardinality alphabet would lose alleles at a faster rate than the equivalent binary coding, even if we used larger initial populations.

To summarize, in order to get the claimed benefit of using higher-cardinality alphabets, we need to use much larger populations than for the equivalent binary-coded chromosomes. We may also need to maintain higher mutation rates in order to compensate for an allele loss rate which is expected to be greater than in the binary case. Finally, we note that this argument supports the use of binary coding *where it is meaningful*. As is pointed out in [Antonisse, 1989], if the q-ary alphabet is *not* a power of 2, an equivalent binary encoding would define some alleles that have no meaning in the higher-cardinality alphabet.

4 SYSTEMATIC SELECTION OF AN INITIAL POPULATION

Given the rather 'large' minimal population sizes needed for the case $q > 2$, we now consider alternative ways of specifying an initial population. The above analysis is of course predicated on the assumption that we choose our initial population as a random sample from all possible q^L strings of length L. The experimental design context would suggest that this is wasteful—rather we should generate the initial population in such a way that it 'covered' the search space in a more systematic way than is likely to occur by a random process. Not only could we thereby ensure that all alleles (i.e. all order-1 schemata) are present in the initial population, but we would intuitively expect the number of schemata of orders higher than 1 to be increased also.

4.1 ERROR-DETECTING CODES

We might expect that the average 'distance' of the optimal point from the initial population of strings would be smaller in the case of a systematically selected population. This points us in the direction of a possible mechanism for carrying out such a selection. The question of how to choose a set of strings using as a criterion some function of their distance from an arbitrary string is one which arises in digital communications engineering. This has led to the development of error-detecting (and error-correcting) codes. (A very brief introduction to code generation is given in the appendix to this paper. However, readers should realize that there is a vast literature on this subject; to find out more, they should look at a thorough treatment such as that given in [Michelson and Levesque, 1985], or [MacWilliams and Sloane, 1977].)

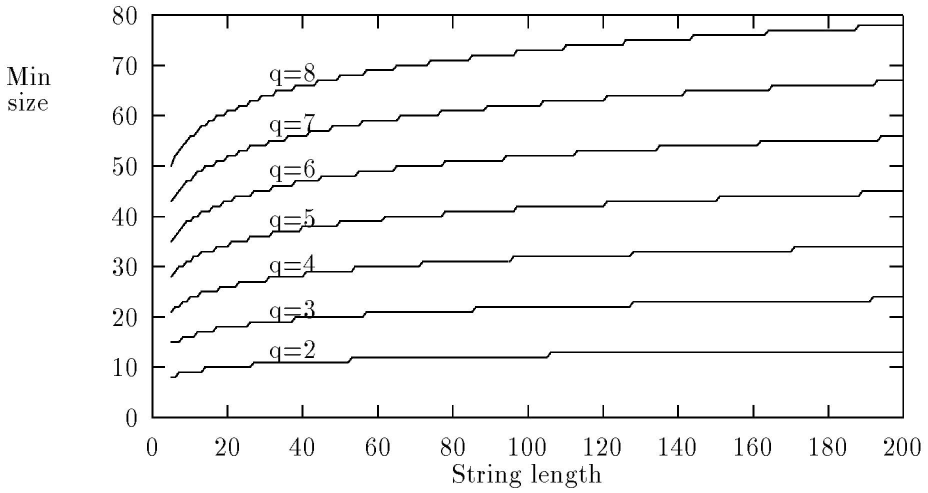

Figure 1: Minimal population sizes for 95% confidence of all alleles present

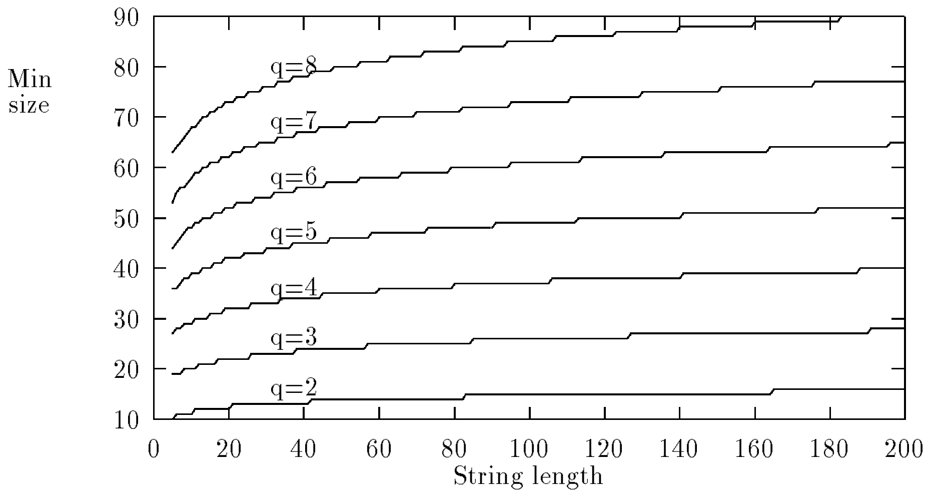

Figure 2: Minimal population sizes for 99% confidence of all alleles present

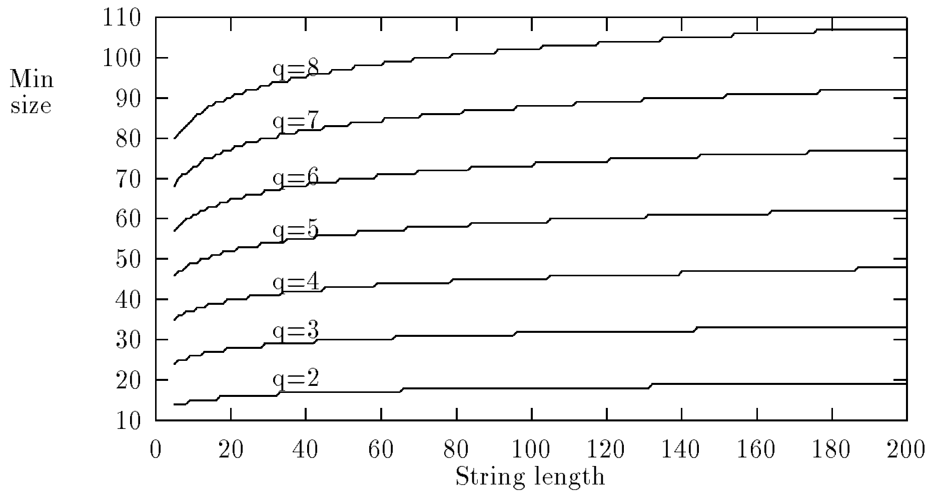

Figure 3: Minimal population sizes for 99.9% confidence of all alleles present

The purpose of a code is usually to encode a message (usually comprising binary digits) k bits long by adding $(n-k)$ bits which have some meaning relative to the actual information (e.g. they may contain a parity check). The information bits plus check bits make up a codeword; the set of all possible codewords is known, and has to satisfy certain criteria. On receiving an incorrect codeword, the actual message may still be deduced by finding the nearest correct codeword to that actually received. This implies that codewords should not be 'close' to each other, but should maintain a minimum distance from each other. Codes are often classified as (n, M, d) where n is the length of the codeword, M is the number of words, and d is the minimum distance. For example, the following is an (11,11,6) code

$$\begin{array}{ccccccccccc}
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{array}$$

We carried out some computations to compare this structured population set with the conventional approach of randomly generating a population. We randomly generated a population of 11 strings, each of 11 bits, and counted the number of schemata of all orders occurring in this population. This procedure was repeated 20 times, in order to calculate some statistics; these are compared in table 1 below with the number of schemata represented by the above (11,11,6) code.

Table 1: Number of schemata present in random and structured populations

Schema order	Number of schemata	Number in random pop. min	mean	max	(11,11,6) code
1	22	22	22	22	22
2	220	202	211	219	220
3	1320	933	1013	1099	1265
4	5280	2433	2684	2905	3355
5	14784	3976	4373	4676	5027
6	29568	4364	4727	4954	5082
7	42240	3310	3509	3610	3630
8	42240	1723	1788	1814	1815
9	28160	590	601	605	605
10	11264	120	121	121	121

It can be seen that, while the difference is marginal for the shorter and longer schemata, there are appreciably more schemata of orders 3-5 in the (11,11,6) code than in even the best random population; in the average case there are at least 20% more schemata of these orders in the systematic sample. It would appear that the initial population has been significantly enriched by selecting a systematic rather than a random sample of strings. This may not be an important effect in conventional GA implementations where many generations can be tested, but we would expect this approach to result in improvements in the context we have outlined above, where the number of possible experiments is limited.

4.2 EXPERIMENTAL DESIGN

The question of 'covering' the search space effectively is one which has also long occupied the minds of statisticians, and in the field of experimental design, a vast array of methods have been developed for this purpose. As an example of an experimental design, we instance the following 'L-16' design for selecting a population of 16 strings each of 15 bits.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 1 1 0 0 1 1 1 1 0 0 1 1 0 0
0 1 1 1 1 0 0 0 0 1 1 1 1 0 0
0 1 1 1 1 0 0 1 1 0 0 0 0 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 1 0 1 0 1 0 1 0
1 0 1 1 0 1 0 0 1 0 1 1 0 1 0
1 0 1 1 0 1 0 1 0 1 0 0 1 0 1
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
1 1 0 0 1 1 0 1 0 0 1 1 0 0 1
1 1 0 1 0 0 1 0 1 1 0 1 0 0 1
1 1 0 1 0 0 1 1 0 0 1 0 1 1 0
```

A similar experiment to that performed above was conducted for this design; the full results are omitted for reasons of space, but in Table 2 we again see that the L-16 design has 10-20% more 'middle-order' schemata than the average of 20 random populations.

Table 2: Number of schemata present in random and structured populations

Schema	Number of	Number in			L-16
order	schemata	random pop.			design
		min	mean	max	
1	30	30	30	30	30
2	420	412	417	420	420
3	3640	3139	3228	3332	3500
4	21840	13607	14177	14953	17640
5	96096	37215	38557	40576	45528
6	320320	68683	71692	74594	79240
7	823680	92711	97346	100141	102840

While the underlying philosophy driving these designs is different from that of error-detection, the methodologies have much in common. Nevertheless, the fact that the experimental design context is not so exclusively concerned with binary representations means that there are also many designs for q-ary alphabets. It is of course those cases where $q > 2$ in which a systematic selection of the initial population can be expected to be most beneficial, bearing in mind some of the minimal population sizes derived above. The results of some preliminary computational experiments do indeed suggest that this is the case; a report [Reeves and Wright, 1993] on these findings is in preparation.

5 CONCLUSION

This paper has considered some theoretical background to the problem of using GAs with small populations and a limited number of fitness evaluations. We have described an approach to specifying a minimal population size for the application of GAs. This has also tended to confirm (*pace* the arguments in [Antonisse, 1989]) that the GA community's instinct for preferring a binary encoding where possible and sensible is soundly based on the dynamics of population size and string length.

Finally, we have demonstrated that, at least for small population applications (and possibly more generally in the case of q-ary alphabets for $q > 2$), it may be important to consider selecting an initial population in a systematic way rather than the conventional random selection procedure, in that there will initially be more schemata available for processing from a systematic selection. We have also suggested ways in which such a selection might be carried out by drawing on procedures developed elsewhere in the generation of error-correcting codes and experimental designs.

We are currently engaged in experimental work to test the effectiveness of these ideas particularly in the context of higher-cardinality alphabets.

A APPENDIX

A.1 A BRIEF INTRODUCTION TO CODE GENERATION

Unfortunately, the generation of codes, although not difficult, is not particularly straightforward either. The many books on coding theory list a variety of codes of particular interest, but usually none of these is appropriate for the particular L (usually denoted by n in a coding-theory context) and M that we require!

Code generation in general can be approached in many ways; here we sketch one possible route. The simplest group of codes, and the most commonly used in practice, is the class of *linear* codes—that is, for a message of length k, there is a set of k linearly independent codewords (known as the *generator* set) from which the whole code can be constructed by a series of linear operations. Within this class, there is a sub-class of *cyclic* codes which allow the easy construction of a generator set.

For cyclic codes, we need an initial generator codeword, which is shifted cyclically $k - 1$ times to form a generator set. The rest of the code can then be found by taking linear combinations of the generator set. The only problem remaining is to determine the initial codeword. It turns out that these are closely related to the theory of *Galois fields*. Codewords can be represented as polynomials in X where X^m represents the m^{th} bit of the codeword (it is conventional to count from bit 0 to $n-1$, rather than from 1 to n). The cyclic Hamming codes, for example, can be generated from a codeword represented by a *primitive* polynomial in the Galois field. Extensive tables of such polynomials are available in the coding-theory literature.

Sometimes, it may be convenient to use a value $n \neq L$ to define a code, which can then either be *extended* (if $n < L$) by adding bits, or *shortened* (if $n > L$) by deleting bits. Similarly, if there are too many or too few codewords, we can *expurgate* the code by throwing away a subset of codewords, or *augment* it by adding a new codeword (often the all 0*s* or all 1*s* codeword).

There are many other possibilities; *non-linear* codes may allow the generation of more codewords for a given minimum distance d than a linear code of the same length. The (11,11,6) code referred to in the main body of the paper is such a code—it belongs to a class known as *symmetric 2-designs.* Other non-linear codes can be generated from *Hadamard matrices*, which, as GA theorists will know, have close connections with Walsh functions.

Finally, we would re-iterate that this is the most basic of introductions to coding theory, and refer interested readers to the literature.

Acknowledgement

The author would like to acknowledge the assistance of his colleague Howard Jones in explaining the basic combinatorics of codes and designs.

References

[Alander, 1992]
J.T.Alander (1992) On optimal population size of genetic algorithms. *Proceedings of CompEuro 92*, 65-70. IEEE Computer Society Press.

[Albrecht *et al.*, 1993]
R.F.Albrecht, C.R.Reeves and N.C.Steele (Eds.) (1993) *Proceedings of an International Conference on Artificial Neural Networks and Genetic Algorithms.* Springer-Verlag, Vienna.

[Antonisse, 1989]
J.Antonisse (1989) A new interpretation of schema notation that overturns the binary encoding constraint. *In* [Schaffer, 1989], 86-91.

[Goldberg, 1985]
D.E.Goldberg (1985) *Optimal initial population size for binary-coded genetic algorithms.* TCGA Report 85001, University of Alabama, Tuscaloosa.

[Goldberg, 1989a]
D.E.Goldberg (1989) Sizing populations for serial and parallel genetic algorithms. *In* [Schaffer, 1989], 70-79.

[Goldberg, 1989b]
D.E.Goldberg (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass.

[Goldberg and Rudnick, 1991]
D.E.Goldberg and M.Rudnick (1991) Genetic algorithms and the variance of fitness. *Complex Systems*, **5**, 265-278.

[Grefenstette, 1986]
J.J.Grefenstette (1986) Optimization of control parameters for genetic algorithms. *IEEE-SMC*, **SMC-16**, 122-128.

[Holland, 1975]
J.H.Holland (1975) *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor.

[Keane, 1993]
A.J.Keane (1993) Structural design for enhanced noise performance using genetic algorithm and other optimization techniques. *In* [Albrecht *et al.*, 1993], 536-543.

[Liu, 1968]
C.L.Liu (1968) *Introduction to Combinatorial Mathematics.* McGraw-Hill, New York.

[MacWilliams and Sloane, 1977]
F.J.MacWilliams and N.J.A.Sloane (1977) *The Theory of Error-Correcting Codes.* North-Holland, New York.

[Michelson and Levesque, 1985]
A.M.Michelson and A.H.Levesque (1985) *Error Control Techniques for Digital Communication.* Wiley, New York.

[Reeves, 1993]
C.R.Reeves (Ed.) (1993) *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, Oxford.

[Reeves and Wright, 1993]
C.R.Reeves and C.C.Wright (1993) Genetic design and Taguchi methods: an experimental comparison. (In preparation).

[Schaffer, 1989]
J.D.Schaffer (Ed.) (1989) *Proceedings of 3rd International Conference on Genetic Algorithms.* Morgan Kaufmann, Los Altos, CA.

[Schaffer *et al.*, 1989]
J.D.Schaffer, R.A.Caruana, L.J.Eshelman and R.Das (1989) A study of control parameters affecting online performance of genetic algorithms

for function optimization. *In* [Schaffer, 1989], 51-60.

[Shainin and Shainin, 1988]
D.Shainin and P.Shainin (1988) Better than Taguchi orthogonal tables. *Quality & Reliability Engineering International*, **4**, 143-149.

[Syswerda, 1989]
G.Syswerda (1989) Uniform crossover in genetic algorithms. *In* [Schaffer, 1989], 2-9.

[Taguchi, 1986]
G.Taguchi (1986) *Introduction to Quality Engineering*. Asian Productivity Organisation, Tokyo.