

Genetic Algorithm Encoding Probabilities & Population Size

Dermot W. O'Dwyer¹ & Eugene J. O'Brien²

1. *Department of Civil, Structural, and Environmental Engineering, Trinity College Dublin, EMail: dwodwyer@tcd.ie*
2. *Department of Civil Engineering, National University of Ireland, Dublin, EMail: eobrien@ucd.ie*

Abstract

Genetic algorithms offer a potentially powerful tool for tackling optimisation problems which cannot be formulated as mathematical programming problems. This paper explores why success in the application of genetic algorithms has not been universal.

The paper first categorises optimisation problems and identifies features of optimisation problems which identify the problems as being suitable for solution using genetic algorithms.

The paper shows how choice of coding system affects convergence and gives guidance on choosing a coding system. The paper also details the relationship between population size, critical schema length, problem constraints and convergence.

This paper examines mutation probability and an expression is developed which gives the optimum rate of mutation. The paper shows that mutation will not reintroduce "long" schema which have been lost from the population. Crossover probabilities are also considered and the effects of crossover are described. An expression is developed which shows the mechanism by which crossover combines short schema to form longer schema. These analyses highlight the sequential nature of evolution program convergence and support the "building block" hypothesis. A modified crossover operation is suggested.

1 Introduction

Genetic algorithms are comparatively recent optimisation techniques which attempt to find optimal solutions to design problems by mimicking the mechanism of natural selection. When a new technique is developed it is often applied widely initially. This has been the case with genetic algorithms. Unfortunately the successful application of genetic algorithms has been less widespread. Difficulties also arise from the variety of genetic algorithms proposed and their problem specific bias. Even where Goldberg's¹ standard Simple Genetic Algorithm is concerned there is a profound lack of guidance on how problems should be coded and what values should be adopted for population size, mutation rate and crossover probability.

This paper addresses these questions and gives guidance on how the problem should be coded. The paper also derives an equation for the optimum mutation probability and develops an expression which sheds light on the effects of single point crossover. The paper investigates the inter-related subjects of constraints, critical schema length and population size.

The expressions derived in this paper are developed for Goldberg's Standard Genetic Algorithm with single point crossover, linear scaling and stochastic selection with replacement. These formulae may need modification before being applied to other genetic algorithms or evolution programs.

This paper broadly accepts the validity of the schema analysis developed by Holland² and others and assumes that the building block hypothesis has validity.

2 Problems Suitable for GA

One fundamental problem which should be avoided with genetic algorithms is applying them to problems to which are unsuited. Therefore it may be helpful therefore to describe the characteristics that would make a problem ideal for solution using genetic algorithms.

1. The objective of an ideal problem would be to find a competitive solution. Although attaining the global optimum is desirable it should not be vital.

2. The problem should be large enough to make any attempt at an exhaustive search futile.
3. The problem should not be a mathematical programming problem. If all the variables are continuous and the objective function is known then the available gradient information should make traditional gradient based techniques more efficient.
4. It should be possible to express the problem using a coding system which allows convergence via the building block hypothesis. Any problem which is very constrained, where there are underlying and unknown constraints, will be difficult to solve.

2.1 Encoding the problem

The importance of the coding system chosen is difficult to overstate. The same problem will either yield to a genetic algorithm approach or prove unsolvable depending on the way in which the problems variables, constraints and objectives are encoded.

This is best illustrated with an example which shows the problems inherent in choosing a coding system for even a very simple problem. Consider the following problem; pick six points on the circumference of a circle such that the points are equally spaced. The problem is trivial and some potential solutions are illustrated in Figure 1 below.

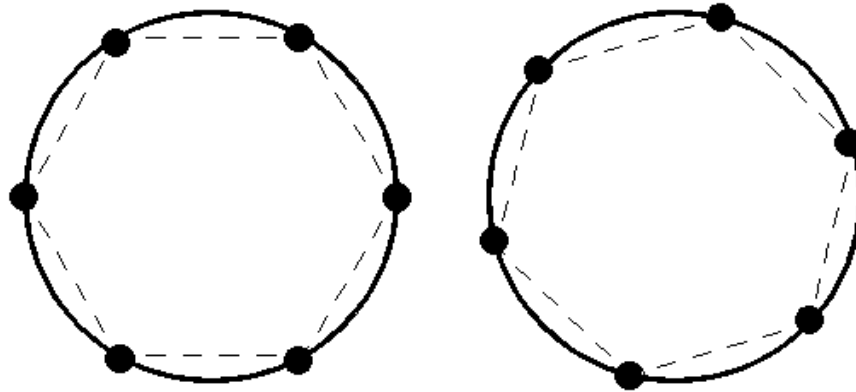


Figure 1. Different optimal solutions.

The location of each node can be described in terms of its angle position, by the length of the arc from some reference point. Ideally each point

should be positioned 60° apart from each of its two nearest neighbouring points as per the two examples in Figure 1.

One simple objective function for this function is to sum the absolute difference from the correct relative separations. For example, if a potential solution were,

$$(21, 37, 49, 138, 290, 342)$$

Then the value of such an objective function would be,

$$242 = \left\{ \begin{array}{l} |60 - (37 - 21)| \\ + |60 - (49 - 37)| \\ + |60 - (138 - 49)| \\ + |60 - (290 - 138)| \\ + |60 - (342 - 290)| \\ + |60 - (360 + 21 - 342)| \end{array} \right.$$

This is not the only possible objective function. Another option would be to square the difference between the actual angular separations and the ideal angular separations.

$$14050 = \left\{ \begin{array}{l} (60 - (37 - 21))^2 \\ + (60 - (49 - 37))^2 \\ + (60 - (138 - 49))^2 \\ + (60 - (290 - 138))^2 \\ + (60 - (342 - 290))^2 \\ + (60 - (360 + 21 - 342))^2 \end{array} \right.$$

Both objective functions will have zero value for an optimal separation and each will penalise sub-optimal solutions according to how far they are from optimal. In the first objective function the penalty will be linear in the second quadratic and hence will apply more selective pressure than the first.

The encoding system given above is not the only possible system. It is important to recognise that there is more than a single way to encode

the nodal positions. The simplest coding system identifies the nodal positions by giving their absolute position, this is the coding system used above. However consider what happens when two such codes are mutated or crossed. For clarity the genetic string parameters are represented directly rather than using binary, thus;

$$\begin{array}{ccc} (020\ 090\ 165\ * \ 240\ 300\ 355) & & (020\ 090\ 165\ 160\ 220\ 300) \\ * & \Rightarrow & \\ (005\ 050\ 100\ * \ 160\ 220\ 300) & & (005\ 050\ 100\ 240\ 300\ 355) \end{array}$$

The first problem is that under either mutation or crossover (as shown above) the nodes will not necessarily remain in order. This problem is easily rectified. There are some simple solutions to this; first the binary or Gray code defining the parameters can be decoded and the parameters arranged in order at which point the binary code can also be rewritten to reflect the reordering.

Rewriting the code at each evaluation seems a drastic step which alters if not undermines the schema theorems, since reordering may translate the schema positions. However, consider the difficulties which arise if the coding is not rewritten. The solutions (060 120 180 240 300 360) and (360 300 240 180 120 060) are both optimal, however if their genetic code is not reorganised so that the magnitude of the parameters is in the same order consider what happens if their strings are crossed. Even in the cross occurs between the schema which define separate nodal positions both resulting strings will be poor. For example consider the potential result (060 120 240 180 120 60) and (360 300 180 240 300 360). Both of these solutions are poor. Clearly this encoding is flawed and the convergence will be very poor.

Even if the binary code is rearranged this encoding has difficulties. For example consider the following two solutions, (030 090 150 210 270 330) and (015 075 135 195 255 315). Both these solutions are optimal, the number of optimal solutions depends on the number of digits used to identify the nodes' positions. Thus if the positions of the nodes are specified to an accuracy of 1° then there will be 360 optimal solutions. This presents problems because single location schema such as ($* \ * \ 135 \ * \ * \ *$) are neither good nor bad but their fitness depends on other parameters in the string. Thus the critical minimum schema length, of which more will be said in section 5.0, will be twice the length for a single parameter. However the problem is more difficult still because although the schema ($* \ * \ 135 \ 195 \ * \ *$) and the schema (60 120 $* \ * \ * \ *$)

are both fit the schema (60 120 135 195 * *) is poor and hence the building block hypothesis will fail and convergence will be poor.

One easy approach could be to fix the position of the first node ,i.e. (30 * * * * *). This would then mean that the schema (30 90 * * * *) would propagate and after this schema had become common in the population then the schema (30 90 150 * *) would prosper and so on. However, there are two disadvantages to this system, first for some reason we might not want all the individuals in the population to converge to a single instance of the many optimal solutions. Second the convergence will probably be slow since the convergence is sequential. First the second parameter is refined then the third then the fourth etc. A better solution would be have the first parameter as the first node's position and define the other five parameters as the angle of arc between adjacent points. In this way the schema (* * 60 * * *) and (* * * 60 * *) can be combined successfully with crossover and will both be propagated immediately, thus realising the genetic algorithms potential for parallel processing.

If a flawed encoding system which does not support the building block hypothesis is used it will converge fastest, if it converges at all, if mutation is used without crossover. The choice of coding has a profound effect on the rate of convergence. This should be remembered when assessing recommendations on mutation and crossover probabilities which were developed by observing the rates of convergence for test functions.

An efficient encoding system must avoid the problems identified. The following guidelines should be considered when developing an encoding system.

- Assume that the building block hypothesis holds
- Select an encoding system which facilitates the building block hypothesis
- Arrange the coding so that where possible the “meaning” associated with a particular bit does not vary throughout the population. Similarly avoid coding systems where the bits controlling a certain property migrate throughout the string.
- Minimise, where possible, encoding systems where the meaning of a particular bit is dependent on the status of other bits or schema. In practice this is difficult.

- Avoid encoding systems which can the represent the same solution with different codes.
- Consider how is the solution likely to progress and identify any obvious potential for convergence to a local optimum

3 Population Size

The size of the population is crucial to a genetic algorithm. However for different problems the size of population required varies. Indeed the population size needed may vary during the course of the optimisation.

The schema theorem and building block hypothesis suggest that genetic algorithms work by testing schema, propagating the schema according to how well they perform, mixing successful schema using crossover and relying on mutation to introduce new schema. In the light of this consider the following questions. First, what happens if the smallest schema which makes “sense” is more than three or four bits long? Second can mutation be relied on to introduce or reintroduce large schema into a population? Third will the building block hypothesis outlined above find the optimum?

Considering the first question gives a valuable insight into the relationship between convergence and population size. Suppose the schema (****10*01*****001******) is particularly good whereas the schemas (****10*****001******), (*******01*******) and (*******001*******) show no particular promise. The probability of the schema (****10*01*****001******) prospering and being propagated depends on the probability of its being sampled. Traditional GA schema theory assumes that all schema are sampled however this can only be the case where the population size is larger than 2^l where l is the string length. In practice not all schema are sampled.

The probability of any particular schema being sampled in the initial population is directly dependent on the schema’s length. As schema length increases so too does the number of other possible schema. The table below gives the number of different schema of length n . For example in the case of any five bits there are 32 different schema while there are 128 different ways in which seven bits can be filled.

$$\begin{array}{rcl} 2^1 & = & 2 \\ 2^2 & = & 4 \end{array}$$

$$\begin{array}{rcl} 2^3 & = & 8 \\ 2^4 & = & 16 \\ 2^5 & = & 32 \\ 2^6 & = & 64 \\ 2^7 & = & 128 \end{array}$$

Therefore in our example the good schema is only one of 128 possible. Our good schema may occur in combination with other fatal schema therefore for it to have a reasonable chance of surviving it should be sampled a number of times. If the population size is too small then propagation of the schema is dependent on chance. The probability of a particular k bit schema being sampled in a population of n_{pop} is given by,

$$1 - \left(1 - \frac{1}{2^k}\right)^{n_{pop}} \quad (1)$$

Which for a schema length of 7 and a population size of 50 give the probability that the particular schema will be included among the initial population at 0.324.

3.1 Bottlenecks and Scaling

A somewhat similar effect to that outlined in section 3.0 can occur if any population goes through a “bottleneck”. In a simplistic sense the success of a genetic algorithm optimisation involves balancing the selective pressure. If the selective pressure is insufficient the algorithm will be slow to converge. However, if the selective pressure is too great then schema will be lost from the population and the algorithm is likely to converge, albeit rapidly, to a local optimum.

If at some stage during the optimisation the members of the current population are all descended from a handful of individuals in the previous population then subsequent populations echo the traits of this handful while traits missing from the reduced population are lost. The “bottleneck” effect can be seen in nature and hence its occurrence in genetic algorithms is not surprising. The population of some of the south sea islands of the Pacific are descended from a small number of people, the passengers of a single canoe perhaps. As a result some of these island populations have very high incidences of certain hereditary illnesses simply because one of the original inhabitants was a carrier.

It is important therefore to ensure that the number of individuals being reproduced from a population does not fall too sharply. If the selective pressure is too great then there is a risk of losing too many schema. This is the rationale behind scaling the fitness function, Goldberg¹, Michalewicz³ and Mitchell⁴, so that the selective pressure is not too great. However the effect can occur even where the maximum selective pressure is limited and a maximum of two copies of the best individual is reproduced in the next population. This effect is particularly likely to occur in initial populations where there are only a few individuals which are feasible.

4 Mutation Probability

Considering mutation without crossover and assuming a mutation probability of p_m , the probability that a specific n bits only will be changed by mutation is,

$$p_m^n (1 - p_m)^{l-n} \quad (2)$$

where l is the total number of bits in the string. If a genetic algorithm converges to a solution which differs from the optimum by only one or two bits, or if an important schema has been lost from the population what value of p_m should be chosen to maximise the likelihood of the necessary changes being introduced through mutation?

For a given problem formulation the value of l will be fixed. The optimum value for p_m can be obtained by differentiating the eqn (2) with respect to p_m and equating the result to zero. This gives,

$$\begin{aligned} \frac{d(p_m^n (1 - p_m)^{l-n})}{dp_m} &= (-1)p^n (l-n)(1 - p_m)^{l-n-1} + np_m^{n-1} (1 - p_m)^{l-n} = 0 \\ &= p_m^{n-1} (1 - p_m)^{l-n-1} (n(1 - p_m) - (l-n)p_m) = 0 \\ &\Rightarrow p_m = \frac{n}{l} \end{aligned} \quad (3)$$

Hence the best probability p_m to make a one bit change most probable is $\frac{1}{l}$ while for a two bit change the optimum probability p_m is $\frac{2}{l}$.

Figure 2 and Figure 3 show the curves for eqn (2) for $n=1$ and $n=2$. The maximum probability is given by eqn (3). While the curves are somewhat similar in shape they are to very different scales. The maximum probability for an n bit change is given by,

$$\left(\frac{n}{l}\right)^n \left(1 - \frac{n}{l}\right)^{l-n} \quad (4)$$

hence if the string length $l = 100$ then the maximum probability of a one bit change is 0.00370

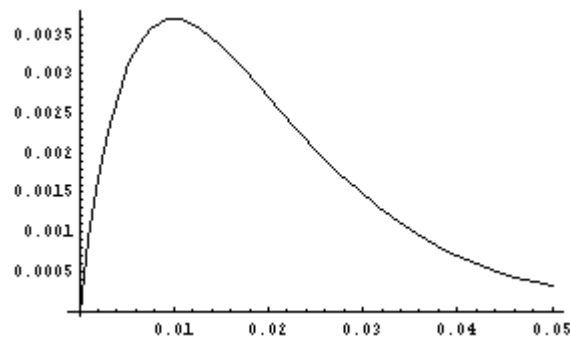


Figure 2. Probability of a particular single bit mutation for different mutation probabilities

while the maximum probability of a specific 2 bit change occurring is 0.00055

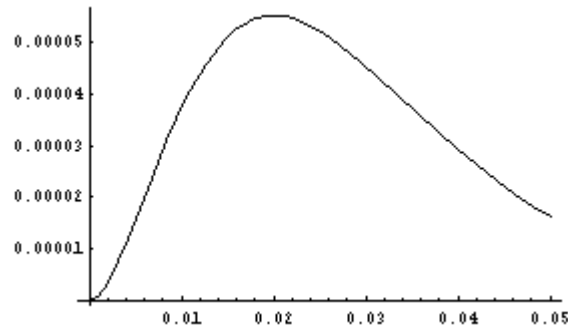


Figure 3. Probability of two particular bits being mutated as a function of the mutation probability

and the probability of a specific 3 bit change occurring is 0.0000014 . These figures show that for a population size of 1000 one could expect between three and four occurrences of the favoured 1-bit-change mutation every generation, 1 occurrence per two generations for the 2-bit change while the favoured three-bit mutation will occur only once every 700 generations.

The purpose of mutation is to introduce or re-introduce short schema. The analysis above indicates the optimal mutation rates with this purpose in mind. However mutation has the negative effect of disrupting good schema. If the optimum has been found, propagated, and survived crossover, or been introduced by crossover then it must survive mutation to be introduced into the next generation intact. The probability of this occurring is,

$$(1 - P_m)^l \quad (5)$$

Figure 4 below shows this curve (with the curve shown in Figure 2 amplified by 100 superimposed upon it) for a string length of 100, for different values of mutation probability P_m .

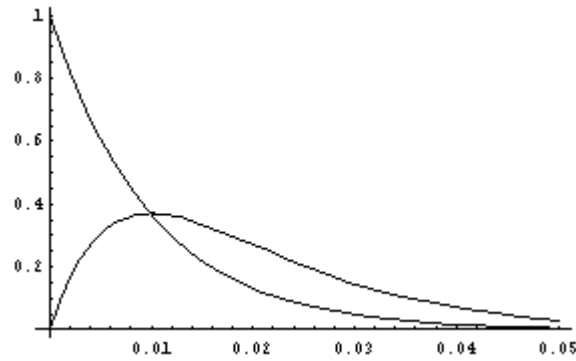


Figure 4. Probability of zero mutations and probability of a particular single bit mutation plotted to different scales.

From Figure 4 one can clearly see that rate at which single bit mutations are introduced is relative insensitive to small changes in mutation rate, whereas the survival of long schema is. Therefore best value for P_m mutation may be somewhat less than the optimal suggested by eqn (3). Initially during an optimisation the length of the schemas should be small but as the algorithm converges the short schemas will be combined to form longer schemas. Therefore there is an argument for P_m to be $\frac{1}{l}$ initially but to be reduced as the algorithm converges.

4.1 Gray's code

The previous section demonstrated that if a solution is dependent on mutation introducing schema longer than one or two bits then the solution will not be found. This analysis also favours the use of a Gray's code rather than the tradition binary code since with Gray's code all adjacent points in the search space differ by a single bit. With a traditional binary code some adjacent nodes will differ by many bits.

Decimal	Traditional Binary	Gray's Code
1	000	000
2	001	001
3	010	011
4	011	010
5	100	110
6	101	111
7	110	101

8

111

100

For example the change from 3 to 4 requires the change of all bits in the binary representation. Thus if an optimisation converged on 3, which in some hypothetical problem might be good but slightly less good than 4, then with a Gray's code mutation may introduce the single bit change required. With a binary representation the necessary simultaneous 3-bit change is less probable

5 Crossover

While crossover is perhaps the most important standard operator in genetic algorithms its effects are not fully understood. There are different ways of implementing crossover from single point and two point crossover to parameterised crossover. However the following analysis concentrates on single point crossover for the simple reason that this is the most basic type of crossover and as such is the easiest to examine. A rigorous examination of crossover is important because although many authors, Michalewicz³, Mitchell⁴, suggest a value of approximately 0.7 for P_c all base their recommendation on empirical evidence. The notation used in the following section is as per Goldberg¹ however the formula resulting differs substantially.

Consider the following simple model. In a given population certain traits confer an advantage on the individuals which have them. To make visualising the problem easy we can label these traits as having red hair and having green eyes. The most fit individuals are those containing the schemata for both red hair and green eyes, H_{RG} . Thus,

$$\frac{f(H_{RG})}{\bar{f}} > \frac{f(H_{RG}^-)}{\bar{f}} \quad \text{and} \quad \frac{f(H_{RG})}{\bar{f}} > \frac{f(H_{RG}^-)}{\bar{f}}$$

while the fitness of those who have neither red hair nor green eyes $\frac{f(H_{RG}^-)}{\bar{f}}$ is worse still.

If mating in this hypothetical population follows the procedure of haploid single point crossover, as per Goldberg's SGA (simple genetic algorithm), consider how the number of red haired green eyed individuals

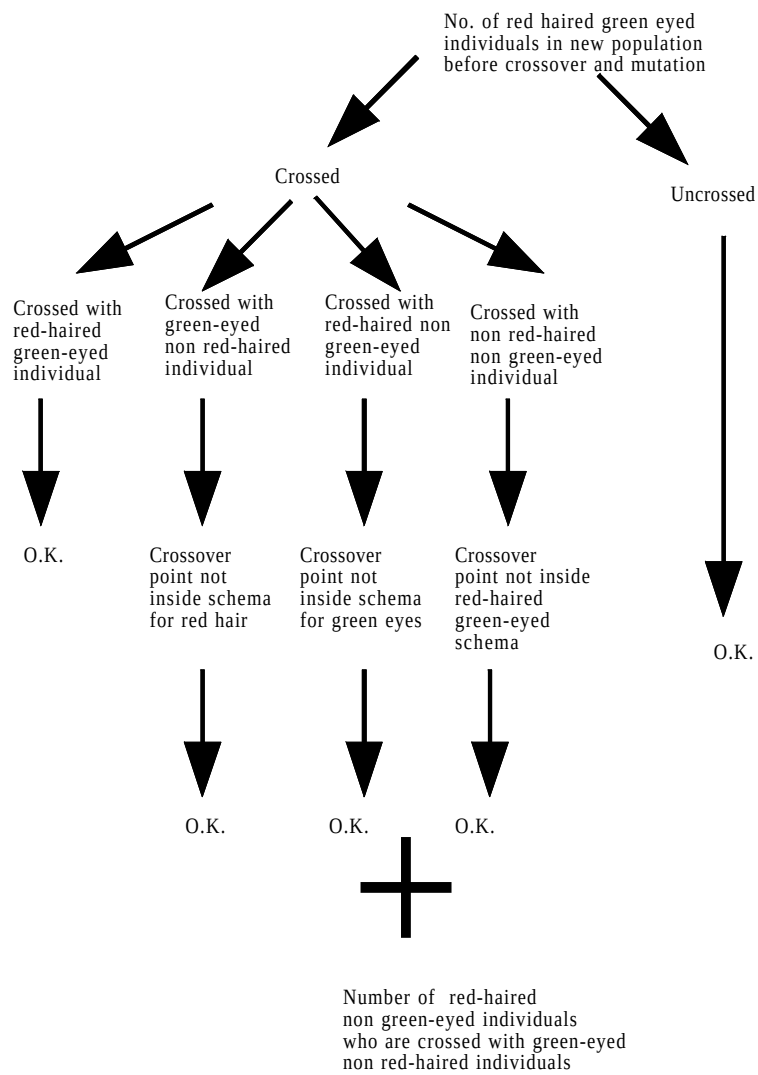


Figure 5. The different ways in which red-haired green-eyed individuals can either survive crossover or be formed by crossover. These individuals must also survive mutation

$m(H_{RG}, n)$ vary from generation n to generation $n+1$. Most, but not all, of the many different ways in which the schema H_{RG} can be propagated from generation n to generation $n+1$ are outlined in Figure 5.

The number of red haired green eyed individuals in the next generation, $m(H_{RG}, n+1)$, will be greater than the number of H_{RG} in the next generation which are not crossed plus those which are crossed in such a way as to leave H_{RG} undisrupted. To these must be added instances of H_{RG} which occur in the next generation through crossing H_{RG}^- and H_{RG}^- favourably. All these instances of H_{RG} must also survive mutation. This tally does not include instances of H_{RG} which occur as a result of mutation.

After some manipulation the following expression for $m(H_{RG}, n+1)$ can be developed,

$$m(H_{RG}, n+1) \geq (1 - P_m)^{d(H_{RG})} \left\{ m(H_{RG}, n) \frac{f(H_{RG})}{\bar{f}} + P_c \left\{ \frac{Term(A) - Term(B)}{npop * \bar{f}^2 * (l-1)} \right\} \right\} \quad (6)$$

Where,

$$Term(A) = m(H_{RG}^-, n) f(H_{RG}^-) m(H_{RG}^-, n) f(H_{RG}^-) [H(RG) - H(G) - H(R)] \quad (7)$$

and

$$Term(B) = m(H_{RG}, n) f(H_{RG}) \left\{ m(H_{RG}^-, n) f(H_{RG}^-) H(R) + m(H_{RG}^-, n) f(H_{RG}^-) H(G) + m(H_{RG}^-, n) f(H_{RG}^-) H(RG) \right\} \quad (8)$$

This expression yields a lot of information. In the first instance since there are no higher powers of P_c in this expression we can conclude that the optimum value for P_c will be either 1 or 0, depending on whether $Term(A)-Term(B)$ is positive or negative. The sign of $Term(A)-Term(B)$ is dependent on; the occurrence of the schemas, the relative fitness of the sub-schemas, and the schemas' lengths and positions. Thus crossover can act favourable on some schema and unfavourably on others simultaneously. Therefore the choice of a value for P_c is a compromise.

Eqns (6), (7) and (8) do at least offer insight as to why crossover is so powerful. If $m(H_{RG}, n) = 0$ and there are no green eyed red haired individuals in the population then $Term(B)$ will equal zero and crossover acts to generate such individuals. In general a substantial number of individuals with the H_{RG} property will be produced through individuals containing the H_{RG} schema being crossed with individuals containing H_{RG} .

Early in the optimisation the number of individuals with red hair and green eyes will be related to the independent occurrence of these traits. If p is the probability of red hair and q is the probability of green eyes then we would expect the probability of an individual having red hair and green eyes to be approximately pq initially. These individuals would be propagated in accordance with their fitness so that, ignoring mutation, the following population should contain approximately

$$pq \times \frac{f(H_{RG})}{\bar{f}} \quad (9)$$

which occurred through this route. If green eyed individuals and red haired individuals are also above average fitness then the number of green eyed red haired individuals which occurs through crossover

$$\begin{aligned} p(1-q) \frac{f(H_{RG})}{\bar{f}} \times (1-p)p \frac{f(H_{RG})}{\bar{f}} \\ = (pq - pq^2 - p^2q + p^2q^2) \frac{f(H_{RG})}{\bar{f}} \times \frac{f(H_{RG})}{\bar{f}} \end{aligned} \quad (10)$$

may be greater than those occurring through direct propagation.

Another consequence of eqns (6), (7) and (8) which is in contradiction with the usual comments on crossover is that for any non-zero value of P_c the schema which are least disrupted and most likely to benefit from crossover are those for which the sub-schema are far apart this follows from the observation that the only positive section in $Term(A)$ is multiplied by $H(RG)$ the length of the schema H_{RG} .

The magnitude of $Term(A)-Term(B)$ can be increased by artificially preventing crossover occurring within $H(R)$ or $H(G)$. In the early stages of the optimisation this might be counterproductive since crossover would also tend to act as a macro mutation operator, this difficulty might be overcome by only limiting the locations where crossover can occur midway through the optimisation. The positions where crossover would be allowed would be between sections of the genetic code which define different parameters.

Eqns (6), (7) and (8) are also affected by scaling the fitness function and any sorting of the data which increases the probability of similar schemas being crossed with each other. However some such manipulations reduce the amount of mixing and limit the introduction of new schema.

It is hoped that once the optimum is found it will continue to be propagated. On that basis, and ignoring the overall effect of the terms detailed in eqns (7) and (8), there is an argument for choosing the mutation probability and maximum scaling factor such that,

$$(1 - P_m)^l \times (Scaling \ factor) \geq 1 \quad (11)$$

6 Conclusions

The successful application of a genetic algorithm is dependent on there being a gradual route towards an optimal solution. The encoding system chosen must aid such a gradual convergence. The selective pressure should not be too high or convergence will be too rapid leading to convergence to a local optimum.

The size of population required to ensure convergence of the algorithm is dependent of the length of the smallest “critical” schema. Put simply the population must be large enough to ensure that all vital schemas are sampled in the initial population.

The mutation probability should be less than or equal to one over the binary string length. The optimum crossover probability will be problem dependent and will vary throughout the optimisation with a high value early in the optimisation and a slightly lower value later. A Gray’s code has some potential advantages over a standard binary encoding.

There are possible advantages to limiting the positions at which crossover can occur during the later stages of convergence. Restricting crossover so that it only occurs between the sections of code defining the different problem parameters.

References

- [1] Goldberg, David E., *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison Wesley, 1989.
- [2] Holland, John H., *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.
- [3] Michalewicz, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Edition , Springer, 1996.
- [4] Mitchell, Melanie, *An Introduction to Genetic Algorithms*, MIT Press, 1996.