Towards computing the parameters of the Simple Genetic Algorithm

Roger Jonsson*and Jacek Malec†

March 13, 2001

Abstract

The problem of finding appropriate probablilities for crossover and mutation with respect to resampling may be addressed using the Markov chain model. Our efforts in this direction lead through a simplification of the mixing matrix incorporating both probabilities. In the paper we present the simplification and discuss some of its ramifications. We expect that it may lead to some improvement of the computational properties of the Markov chain model of the Simple Genetic Algorithm.

1 Introduction

In order to apply a genetic algorithm (GA) to solve a real-world problem one has to decide which representation to use, how to construct the fitness function and which parameter values to apply: all those decisions affect the efficiency of search. In this paper we focus on a specific GA model, namely the Simple Genetic Algorithm [11], and two of its crucial parameters, i.e., probabilities of crossover and mutation. Normally these parameters are set by the users on the basis of their experience and knowledge about the problem at hand. Even then, the parameters usually have to be further tuned in order to obtain good performance. Some work has been done in the area of finding suitable parameter values, but the results have been rather inconclusive [2, 7].

Some years ago the No-Free-Lunch Theorem [12] shed new light on this problem. If we assume that the crossover and mutation probabilities differently affect the performance depending on the problem, then there are no "perfect" parameter settings. Whether this assumption is valid or not depends on what is meant by "performance". Here we understand performance to be the smallest number of chromosomes that the GA needs to consider in order to find with a given probability a specific individual or group of individuals (e.g., the optimal

*Department of Computer Engineering, Mälardalen University, P.O. Box 883, 721 23 Västerås, Sweden, email: roger.jonsson@mdh.se

†Department of Computer Science, Lund University, P.O. Box 118, 221 00 Lund, Sweden, email: jacek@cs.lth.se

one(s)). With this understanding it is clear that resampling (i.e., reconsidering the same chromosome again) reduces the performance and should, if possible, be avoided.

Within self-adapting GAs (SAGA) these parameters are tuned more or less automatically. Regardless of how these self-adaptive genetic algorithms regulate the parameters they all have a common objective: To tune the GA so that it finds a good enough solution as fast as possible. Some SAGAs adjust the mutation probability depending on time in order to prolong the evolution (i.e., to avoid the premature convergence); see e.g. [4]. Some other use feedback from the population for that purpose [5]. A good overview of different SAGA methods can be found in [3] and different desired behaviours of SAGA are described in [1].

Spears and De Jong have explained with their disruption/construction theory (recently summarized in [9]) how mutation and crossover affect schemata. Although this theory gives interesting insight in behaviours of these operators, it does not reveal how their probabilities should be set.

The closest work to ours is the one by Suzuki [10]. He has reported some results based on Markov Chain analysis regarding the influence of the mutation distribution on the behaviour of a GA. There were though some problems with calculating the correct probability distribution with respect to the given problem.

The aim of this research is to investigate if and how the probabilities of mutation and crossover affect the probability of resampling. Also, with the above understanding of performance, we expect to derive how these parameters should be set for a specific problem in a given domain.

The paper begins with a short recapitulation of the Nix and Vose's Markov chain model of the SGA. After specialising the model (in Section 3) an interesting simplification is derived in Section 4. Finally we comment this result and give some conclusions.

2 Preliminaries

This section introduces the Simple Genetic Algorithm (SGA) expressed in terms of a Markov chain. This way we introduce the necessary vocabulary that will be used later in the paper.

The theoretical model that Nix and Vose presented in [6] (recently recapitulated in [11]) is a complete model of a generational SGA, with selection, crossover and mutation. The disadvantage of this model (as pointed out by e.g. Spears [9]) is that it can be reasonably used only for very small populations and small chromosome lengths. This is so because the calculations rapidly get unmanageable, both in terms of time and of memory size.

Let C be the cardinality of the representation and L the length of the individual. Thus we have $V = C^L$ possible individuals in the search space. To model a series of n generations we first define a *population* q as a vector: $q \stackrel{\text{def}}{=} < a_0, a_1, \ldots, a_{V-1} >$ were each a_i, for $i = 0, 1, \ldots, V-1$, corresponds to the proportion of that individual in the population, i.e. $1 = \sum_{i=0}^{V-1} a_i$. Let the size of the pupulation, i.e. the amount of individuals in the population, be denoted by U. The population size will be assumed constant throughout this paper, unless noted otherwise.

Then we define a matrix Q with the transition probabilities between each state, were a state is population at a given time. Let S_t be a random variable defining the state at time t. Then the elements in the matrix Q are defined as:

$$Q(q, r) = p_{qr} \stackrel{\text{def}}{=} P(S_t = r | S_{t-1} = q) \tag{1}$$

The size of the matrix Q is $B * B$, were B grows with the chromosome length L, the cardinality C, and the size of the population U. It may be easily computed as $B = \binom{U+V-1}{V-1}$ (see e.g. [6]).

One may also be interested in finding out probability of reaching a specific state or a set of states J, after n generations. A specially useful example of a set J may be the set of all states containing the optimal (in some specific sense) individual. This probability can be calculated as follows:

$$p_J^{(n)} \stackrel{\text{def}}{=} \sum_{r \in J} \sum_q p_q^{(0)} p_{qr}^{(n)} \tag{2}$$

were $p_q^{(0)}$ is the probability of being initially in state q and $p_{qr}^{(n)} \stackrel{\text{def}}{=} Q^n(q, r)$.

Given an SGA and a problem at hand, how can one compute the elements of the matrix Q? Nix and Vose model the crossover and mutation operators with a matrix M and the selection process with a function $\mathcal{F}$, that incorporates the fitness of the individuals. Then:

$$Q(q, r) = p_{qr} = U! \prod_{z=0}^{V-1} \frac{\left(\sum_{x=0}^{V-1} \sum_{y=0}^{V-1} \mathcal{F}(q)_x \mathcal{F}(q)_y M_{x \oplus z, y \oplus z} \right)^{Ur_z}}{(Ur_z)!} \tag{3}$$

where $\oplus$ denotes logical XOR. The selection function $\mathcal{F}(q)_x$ is interpreted as the probability of selecting x from the population q. Because it is convenient to think of $\mathcal{F}(q)$ as a vector of values, we use the standard subscript notation to denote some particular coordinate of this vector.

The matrix M contains the elements $m_{x,y}(0)$ which are the probabilities of producing an individual with only zeros from the parents x and y. The reason why M does not need to include other children than those with only zeros is that the following can be easily shown: $m_{x,y}(z) = m_{y,x}(z) = m_{x \oplus z, y \oplus z}(0)$. Therefore we may skip the argument and speak just of the elements $m_{x,y}$.

The probability of producing a child z from parents x and y depends directly on the type of the recombination operator and on parameter values that this operator uses. A generic formula can be given though for some special cases. E.g. when first crossover is applied, with χ_k being the probability of using crossover mask k, and then mutation is used, with μ_l being the probability of using mutation mask l, then

$$m_{x,y}(z) = \sum_{l,k} \mu_l \frac{\chi_k + \chi_{\bar{k}}}{2} [x \otimes k \oplus \bar{k} \otimes y \oplus l = z] \tag{4}$$

where $\otimes$ is the binary operator AND[1] and $\bar{k}$ denotes the bitwise negation of k. This equation yields the time complexity $O(LV^2) = O(LC^{L^2})$ when the operators $\oplus$ and $\otimes$ have complexity $O(L)$. $[\alpha = \beta]$ should be read as one if $\alpha = \beta$ and zero otherwise.

The crossover mask is a binary string were a zero corresponds to selecting that gene from one parent and a one to selecting that gene from the other parent. For the case when the cardinality is two, the mutation mask is also a binary string were a zero means "no mutation". When the cardinality is larger than two the operators $\otimes$ and $\oplus$ need a more general definition. Please observe that the index l has two (equivalent) interpretations: as an integer (index) and as a binary vector (the binary representation of the integer).

3 Specialising the Markov chain model

From the general description of an SGA given above we will derive a more specific one by concretely specifying the three operators: rank-based selection, uniform crossover and bit-flip mutation with a given rate. By introducing this specialisation we will make it possible to eventually simplify Equation 4.

Rank-based selection can be described as the probability of selecting individual x from population q:

$$\mathcal{F}(q)_x \stackrel{\text{def}}{=} \int_{\sum_j [f_j < f_x] q_j}^{\sum_j [f_j \leq f_x] q_j} \varrho(y) dy \tag{5}$$

where f_x is the fitness of individual x, and $\varrho(y)$ is any continuous increasing probability density over $[0, 1]$, for example $\varrho(y) = 2y$ (cf. [11]). The factor μ_l for bit-flip mutation that uses the mutation rate p_m is:

$$\mu_l = p_m^{\vec{1}^T l} (1 - p_m)^{L - \vec{1}^T l} \tag{6}$$

where $\vec{1}^T l$ is the result of multiplying a transposed vector of ones with l, i.e., the sum of all elements (in case of the mutation mask they are only ones or zeroes) in l. Since the probability distribution of bit-flip mutation with a given rate only depends on the number of ones in the mask, a simpler factor, δ_i, can be defined as follows:

$$\delta_i \stackrel{\text{def}}{=} p_m^i (1 - p_m)^{L-i} \tag{7}$$

where $0 \leq i \leq L$.

Finally, the factor χ_k for uniform crossover, were p_c is the probability of crossover, may be given as:

$$\chi_k = \begin{cases} p_c 2^{-L} & \text{if } \vec{1}^T k > 0 \\ 1 - p_c + p_c 2^{-L} & \text{if } \vec{1}^T k = 0 \end{cases} \tag{8}$$

Since χ_k is constant in the case of $\vec{1}^T k > 0$, we name that case χ_1 and, respectively, χ_0 when $\vec{1}^T k = 0$.

[1] Although it could be more legible to use the standard notation $\wedge$ for logical AND, we have decided to adhere to Vose's [11] choice of symbols.

4 A Simplification

In this section we present the result of computations that we have done for the M matrix given the specialisation described above.

In order to simplify the notation in the following equations, three terms have to be introduced: the number of ones at the corresponding locations (of individuals x and y), $N = \vec{1}^T(x \otimes y)$, the number of zeros at the corresponding locations, $Z = \vec{1}^T(\bar{x} \otimes \bar{y})$, and the number of equal elements at the corresponding locations, $E = Z + N$. Now, the following result can be shown.

Corollary 1 *Given the binary representation ($C = 2$), the mutation distribution defined as bit-flip with a rate p_m (as described by Equation (6)) and the uniform crossover with probability p_c (as described by Equation (8)), the elements of the mixing matrix M may be computed as follows:*

$$m_{x,y} = \frac{(\delta_{\vec{1}^T x} + \delta_{\vec{1}^T y})(\chi_0 - \chi_1)}{2} + \chi_1 \sum_{i=N}^{L-Z} \delta_i 2^E \binom{L-E}{i-N}. \tag{9}$$

Proof: We will depart from the generic form stated in Equation 4 and by subsequent transformations eventually derive Equation 9.

The derivation begins with substituting $z = 0$ in Equation 4:

$$m_{x,y}(0) = \sum_{l,k=0}^{V-1} \mu_l \frac{\chi_k + \chi_{\bar{k}}}{2} [x \otimes k \oplus \bar{k} \otimes y \oplus l = 0] \tag{10}$$

Remember that $m_{x,y}(0)$ may be shortened to $m_{x,y}$.

If we set $k = 0$ then $x \otimes k = 0$ and $y \otimes \bar{k} = y$ thus $y \oplus l = 0$ yields $y = l$, therefore $\mu_l = \mu_y$. Analogously, if we set $k = V - 1$ then $\mu_l = \mu_x$. Extracting those two cases from Equation 10 we obtain the following term:

$$\frac{(\mu_x + \mu_y)(\chi_0 + \chi_1)}{2}. \tag{11}$$

Please remember that χ consist only of two cases ($k = 0$ and $k > 0$). Setting term (11) in Equation 10 yields:

$$m_{x,y} = \frac{(\mu_x + \mu_y)(\chi_0 + \chi_1)}{2} + \sum_{l=0}^{V-1} \sum_{k=1}^{V-2} \mu_l \chi_1 [x \otimes k \oplus \bar{k} \otimes y \oplus l = 0] \tag{12}$$

In order to restore the uniform summation bounds in the inner sum we introduce yet another term:

$$m_{x,y} = \frac{(\mu_x + \mu_y)(\chi_0 + \chi_1)}{2} - (\mu_x + \mu_y)\chi_1 + \sum_{l=0}^{V-1} \sum_{k=0}^{V-1} \mu_l \chi_1 [x \otimes k \oplus \bar{k} \otimes y \oplus l = 0] \tag{13}$$

After some simple transformations we obtain:

$$m_{x,y} = \frac{(\mu_x + \mu_y)(\chi_0 - \chi_1)}{2} + \chi_1 \sum_{l=0}^{V-1} \mu_l \sum_{k=0}^{V-1} [x \otimes k \oplus \bar{k} \otimes y = l] \tag{14}$$

Now we can substitute μ_l by δ_i, were $0 \leq i \leq L$, since the following equality is true:

$$\sum_{l \in A_j} \sum_{k=0}^{V-1} [x \otimes k \oplus \bar{k} \otimes y = l] \;=\; \sum_{k=0}^{V-1} [\vec{1}^T (x \otimes k \oplus \bar{k} \otimes y) = j] \tag{15}$$

where the set A_j holds all masks l such that $j = \vec{1}^T l$. Thus, in Equation 14 we may substitute μ with δ and get:

$$m_{x,y} = \frac{(\delta_{\vec{1}^T x} + \delta_{\vec{1}^T y})(\chi_0 - \chi_1)}{2} + \chi_1 \sum_{i=0}^{L} \delta_i \sum_{k=0}^{V-1} [\vec{1}^T (x \otimes k \oplus \bar{k} \otimes y) = i] \tag{16}$$

Studying the external summation in the equation above we can see that for the condition $[\vec{1}^T (x \otimes k \oplus \bar{k} \otimes y) = i]$ to be true, i has to be at least equal to the amount of ones in corresponding locations of x and y. That is $i \geq \vec{1}^T (x \otimes y) = N$. Analogously, for the condition to be true i can not be larger than the number of locations not containing zeros in the corresponding locations of x and y. That is $i \leq L - \vec{1}^T (\bar{x} \otimes \bar{y}) = L - Z$, and the motivation is similar to the one given above.

Therefore the bounds on the external sum can be tightened without loss of terms:

$$m_{x,y} = \frac{(\delta_{\vec{1}^T x} + \delta_{\vec{1}^T y})(\chi_0 - \chi_1)}{2} + \chi_1 \sum_{i=N}^{L-Z} \delta_i \sum_{k=0}^{V-1} [\vec{1}^T (x \otimes k \oplus \bar{k} \otimes y) = i] \tag{17}$$

Now consider the case when there are E positions in x and y that hold the same value. Then there must be 2^E different crossover masks (k) that produce the same result. Also if there are E equal values in the corresponding locations, there must be $L - E$ positions were the values differ. The amount of combinations of these $L - E$ positions is $\binom{L-E}{i-N}$, where $N \leq i \leq L - Z$. These two terms replace the inner sum, which completes the proof. $\square$

As can be easily seen from Equation 9, this simplified calculation of one element in the mixing matrix has the worst case time complexity $O(L^2)$.

5 On performance

If we define performance as the inverse of the smallest number of chromosomes that the GA needs to consider in order to find with a given probability a specific individual or group of individuals. Then, in order to maximize performance, the GA should consider each chromosome (individual) at most once, since each time the algorithm resample, the performance necessarily decreases. The probability of resampling is particularly high close to convergence, i.e. when the population is homogeneous. The mutation and crossover probabilities should therefore be chosen appropriately, so that they maximise the performance.

To be able to calculate these probabilities, we must first notice how states in the Markov chain model influence the performance. Obviously, only some states, namely those that do not contain any duplicates, are desired — otherwise resampling necessarily takes place. Given the set of all states of the Markov chain, S, if we define a set, D, as the set of all desirable populations

$$D \stackrel{\text{def}}{=} \{q \in S | q =< q_0, q_1, \ldots, q_{V-1} > \wedge \forall i (q_i = 0 \vee q_i = \frac{1}{U})\} \tag{18}$$

then the amount of states in D is $\binom{V}{U}$. The columns of Q that correspond to the states in D (let's call this part of the transition matrix Q_D) show the probabilities of reaching a desirable population from any population. The elements in these columns describe the probabilities of maintaining (if the present population is included in D) or increasing the diversity from one population to another.

Clearly still some of the elements in Q_D are transitions that decrease the performance. For instance, any transition from a state q to a state r, where both states have a non-zero element at the corresponding position (i.e. $\exists i(q_i > 0 \wedge r_i > 0)$) will necessarily result in resampling, although in the next generation. Therefore we may restrict the transition matrix Q_D even further, to a matrix Q_H such that the target states (i.e. the columns) correspond to those states for which no resampling takes place. A corresponding characterisation of the set H (of even more desirable states) could be:

$$H \stackrel{\text{def}}{=} \{q \in D | q =< q_0, q_1, \ldots, q_{V-1} > \wedge (\forall r \in H)(q \cdot r = 0)\} \tag{19}$$

encapsulating all transitions from q that do not lead to resampling. This reduces the number of states even more and the amount can be calculated as: $\binom{V}{U} - \binom{(V-a)}{U}$, were $a = \sum_{i=0}^{V-1} q_i > 0$. The interpretation is that the second term of the subtraction represents the amount of states that include individuals already present in q.

Finally one would like to be able for a given problem to automatically calculate such p_m and p_c that maximise the probabilities in Q_H. To solve this problem we do not need the complete Q matrix, although the complete M (mixing) matrix and the fitness distribution are required.

6 Conclusions and future work

We have presented a new formula for calculating the elements in the mixing matrix M, of complexity $O(L^2)$, as compared to the general formula which has complexity $O(LC^{L^2})$. This reduction in complexity can be achieved only for the case when a binary alphabet, uniform crossover and bit-flip mutation with a given rate are used. Although the simplification is rather restricted, the binary representation and the chosen operators are the ones used rather commonly.

We have also presented the concept of desirable states and some ideas on how to calculate the transition probabilities to these states. Our hypothesis is that maximising these probabilities by manipulating p_m and p_c may give us knowledge about how to choose these parameters for specific cases.

However, there remains much to be done:

- The complexity of maximising Q_H needs to be considered.
- The matrix Q_H is only a snapshot of one transition. The continuous and limit behaviour of a SGA needs to be investigated separately.
- The size of the matrix Q is critical, since it rapidly gets unmanageable. For example, let us consider $L = 5$, a binary alphabet and $U = 10$. Then, from the equation $B = \binom{U+V-1}{V-1}$ (which calculates the dimension of Q), we get $B > 10^9$. This indicates that it is computationally unmanageable (even after applying an aggregation method suggested by Spears in [8]) to calculate all elements of Q for any real world problems.

Our future work will include explicit solving the maximisation problem for small populations and chromosome lengths. The framework will also be extended with the formal analysis of continuous and limit behaviour of the transitions defined by Q_H. However, for this approach to be useful in practice, the fitness distribution needs to be either estimated or sampled, which is a clear disadvantage.

Acknowledgement

The authors are very grateful to Edyta Szymańska for valuable comments on the draft of this paper.

This research has been supported by the Volvo Research Foundation, Volvo Educational Foundation and Pehr G. Gyllenhammar Research Foundation.

References

[1] H.-G. Beyer and K. Deb. On the desired behaviours of self-adaptive evolutionary algorithms. In *Parallel Problem Solving from Nature - PPSN VI*, pages 59–68, 2000.

[2] K. A. DeJong. *An analysis of the behaviour of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[3] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2), 1999.

[4] F. Herrera and M. Lozano. Adaptive control of the mutation probability by fuzzy logic controllers. In *Parallel Problem Solving from Nature - PPSN VI*, pages 335–344, 2000.

[5] C. W. Ho, K. H. Lee, and K. S. Leung. A genetic algorithm based on mutation and crossover with adaptive probabilities. In *Congress on Evolutionary Computation*, 1999.

[6] A. E. Nix and M. D. Vose. Modelling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[7] J. D. Schaffer, R. Caruana, L. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for functional optimization. In *3rd International Conference on Genetic Algorithms*, pages 51–60, 1989.

[8] W. M. Spears. Aggregating models of evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 631–638, 1999.

[9] W. M. Spears. *Evolutionary Algorithms. The Role of Mutation and Recombination.* Natural Computing Series. Springer Verlag, 2000.

[10] J. Suzuki. A Markov chain analysis on simple genetic algorithms. *IEEE Transaction on Systems, Man, and Cybernetics*, 25(4):655–659, April 1995.

[11] M. D. Vose. *The Simple Genetic Algorithm, Foundations and Theory.* MIT Press, 1999.

[12] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transaction on evolutionary computation*, 1(1), April 1997.