A Review of Models for Simple Genetic Algorithms and Cellular Genetic Algorithms

Darrell Whitley

Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
whitley@cs.colostate.edu

1 Introduction

Genetic algorithms employ a form of simulated evolution to solve difficult optimization problems. Genetic algorithms are best suited to solving optimization problems that cannot be solved using more conventional methods. Thus, they are often applied to problems that are nonlinear with multiple local optima. They can also be useful when more traditional gradient descent algorithms are not applicable.

Genetic algorithms work with a population of strings that encode a problem's parameters. For example to optimize the function F(X,Y,Z) the variables X, Y and Z are coded as a string which serves as an artificial chromosome. In this paper, variables are represented as bit strings. An evaluation function determines the relative performance of any set of parameters. Strings of parameters are selected for reproduction and recombined in order to sample new points in the search space. Strings are probabilistically given the chance to reproduce at a rate that reflects their "fitness" relative to the remainder of the population. Reproductive opportunities are allocated such that the best strings receive more opportunities to reproduce than those exhibiting poor performance.

For example, if each variable X, Y and Z is coded as an unsigned 5 bit binary substring, then the full string 110100110010110 and the string yxyyxyxxyyyyxyxx (where x and y are used to represent 0 and 1) would represent the parameter sets (26, 12, 22) and (22, 19, 20) respectively. A random crossover point is chosen without regard to parameter boundaries and 1-point crossover is applied to produce, for example, the following offspring.

```
11010  \/ 0110010110    ==>   11010yxxyyyxyxx
yxyyx  /\ yxxyyyxyxx    ==>   yxyyx0110010110
```

While there are crossover operators that offer advantages over 1-point crossver (Spears and DeJong, 1991), 1-point crossover has the advantage of allowing for easy analysis, and hence is commonly used in theoretical work.

Genetic algorithms statistically sample the global search space and concentrate search in hyperplanes that are likely to contain good solutions. They do this by using recombination to pass hyperplane information from parent strings to offsprings. For example, under simple recombination children of the strings 10101100 and 11011110 must reside in the hyperplane 1***11*0 because both parents shared these bits in common. The string 1***11*0 is referred to as a *schema* and corresponds to a hyperplane partition which contains all strings generated by enumerating over the * positions in the schema. Selective pressure leads to more fit schema being sampled by more individuals in the evolving population. Theoretically, the sampling rate of highly fit well represented hyperplanes is changed by selection to direct the search toward partitions in hyperspace that tend to contain above average solutions (Holland, 1975; Goldberg, 1989). Recent theoretical breakthroughs include exact models of the canonical genetic algorithm (Vose and Liepins, 1991; Whitley et al. 1992). There has also been recent criticism that the existing theory concentrates too much on the role of selection, and not enough on the scattering effects of crossover, but the analytical models reviewed in this paper currently provide little insight as to how to better understand these phenomena.

In this paper different exact models for the simple genetic algorithm are compared and shown to be equivalent. In particular, we relate Goldberg's (1987) models for 2-bit strings to the Vose and Liepins (1991) models by showing that there is an exact correspondance between the two. The models introduced by Whitley et al. (1992) are intermediate between the Vose and Liepins models and the earlier models introduced by Goldberg. It is also shown that producing either 1 or 2 offspring during genetic search does not impact the outcome of these models when recombining binary strings. Finally, the impact of probabilistic crossover is also examined in the context of the Vose and Liepins mixing matrix. Whitley (1994) presents a tutorial on genetic algorithms which introduces the various models as well as the more basic principles of genetic search.

The parallelization of genetic algorithms would appear to be both a natural step toward further emulating biological evolution as well as an opportunity to exploit novel parallel architectures that are becoming increasingly accessible. Regardless of what type of parallel implementation is used, one of the first issues that must be considered is the role of locality in mating and selection. Should mating and selection be global, as in the case of Holland's (1975) canonical genetic algorithm, or should mating be influenced by locality considerations by restricting recombination such that strings must be in the same local neighborhoods in order to reproduce?

It has often been reported that various forms of parallel genetic algorithms produce superior optimization performance on various test functions (Whitley and Starkweathter, 1990; Mühlenbien, 1991). It has also been speculated that locality and the resulting impact on genetic diversity played a key role in this performance (Whitley and Starkweather, 1990). One common type of parallel genetic algorithm that uses local mating and selection is sometimes referred to as a "Fine Grain" or "Massively Parallel" genetic algorithm. These parallel genetic algorithms typically have individual strings residing at cells (processors) which are often arranged as a grid or a torus; strings select mates and recombine with other strings in their immediate neighborhood. It can be shown that this form of parallel genetic algorithm is a subclass of cellular automata. Thus, the term *Cellular Genetic Algorithm* is proposed to describe any algorithm in which individuals reside at cells in some specific topology and utilize a local form of selection and mating (Whitley, 1993).

2 A Simple Genetic Algorithm

The following description is one version of a typical simple genetic algorithm (Goldberg, 1989). The first step in the implementation of the simple genetic algorithm is to generate an initial random population. Each member of the population is a binary string of length L representing some encoding of a set of parameters. After generating a population, each string is then evaluated and assigned a *fitness* value $f_i/\bar{f}$, where f_i is the evaluation associated with string i and $\bar{f}$ is the average evaluation of the strings in the population.

The execution of the genetic algorithm can be viewed as a two stage process. Selection is applied to the population at time t to create an *intermediate population*, where time is incremented each generation. In one generation, the entire population undergoes selection and recombination. Later in this paper, the vector p will represent the distribution of strings in the population at time t, before selection. The vector s will represent the population after selection, but before recombination. Recombination and mutation are applied to the intermediate population to create the next population at time $t + 1$.

In the first generation the current population is also the initial population. Each string in the population is evaluated by calculating $f_i/\bar{f}$. For each string i where $f_i/\bar{f}$ is greater than 1.0, the integer portion of this number indicates how many duplicates of that string are placed directly in the intermediate population. All strings then place additional duplicates in the intermediate population with a probability corresponding to the fractional portion of $f_i/\bar{f}$. After selection is complete, crossover is applied to randomly paired strings with a probability denoted R_c. The resulting offspring are then inserted into the next population.

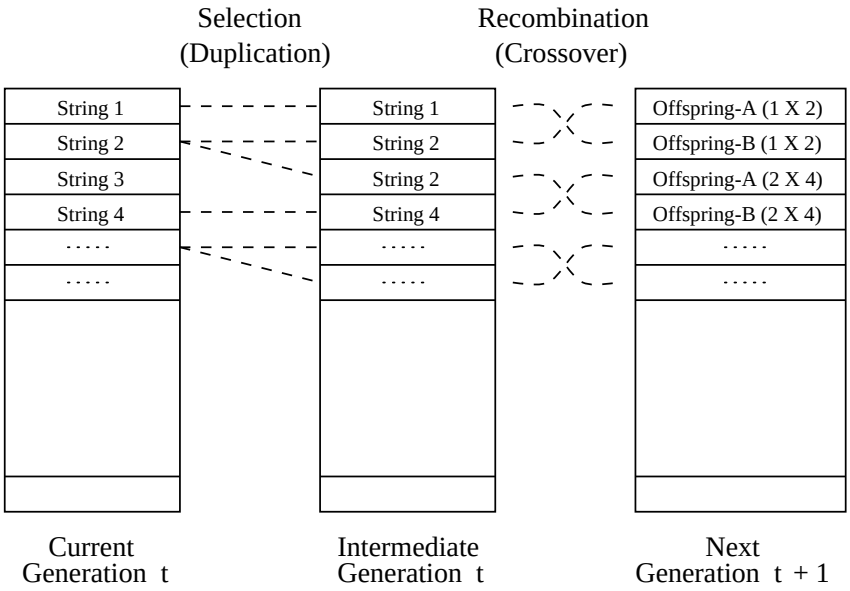

Figure 1: *One generation is broken down into a selection phase and recombination phase. The distribution of strings in the population at time t is characterized by a vector p^t while the distribution of strings in the intermediate population is described by a vector s^t. The next generation is characterized by the vector p^{t+1}.*

2.1 Analytic Models of Simple Genetic Algorithms

Goldberg (1987,1989) introduced the first exact model of the genetic algorithm when processing strings of length 2 in order to analyze the minimal deceptive problem. Bridges and Goldberg generalized this model to look at arbitrary individual strings and schemata (1987); Whitley et al. (1992) further generalized the Bridges and Goldberg model to provide a complete model of how all strings in the search space are processed by a simple genetic algorithm using selection and crossover. Vose and Liepins (1991) independently introduced another exact model which also includes the effects of mutation. In this section, the relationship between the various models is explored. These analytical models are also related to Holland's schema theorem.

Before stating the schema theorem, a generalized form is presented which covers both the schema theorem and the exact models reviewed in this paper.

$$P_H^{t+1} = P_H^t \frac{f_H^t}{\bar{f}}(1 - \{R_c \text{ losses}\}) + \{R_c \text{ gains.}\} \tag{1}$$

where P_H^t is the expected value of the proportional representation of the schema representing hyperplane H in the population at time t. Note that H can also represent individual strings (i.e., a hyperplane partition containing a single string). For hyperplanes, f_H^t is the average evaluation of the strings in the population that are contained in hyperplane H. Note that f_H^t can vary depending on which strings in H are sampled by the genetic algorithm population. For individual strings, f_i does not vary and hence a time index is not required. R_c is the probability of applying recombination. The term *losses* refers to the aggregate probability that strings which currently sample H are recombined with other strings and that the resulting offspring do not sample H. The term *gains* refers the aggregate probability that two strings which currently do not sample H are recombined and that one of the resulting offspring does sample H.

Both the various exact models and the schema theorem are detailed specializations of equation 1. A typical version of the schema theorem is given as follows.

$$P_H^{t+1} \geq P_H^t \frac{f_H^t}{\bar{f}} \left[1 - R_c \frac{\Delta(H)}{L-1}(1 - P_H^t)\frac{f_H^t}{\bar{f}}\right] \tag{2}$$

where R_c is the crossover probability and $(1 - P_H^t)f_H^t/\bar{f}$ represents the proportion of the population that does not fall in hyperplane partition H *after selection has occurred.* When 1-point crossover is used $\Delta(H)/L - 1$ is the probability that crossover will fall within the significant portion of a schema during crossover: this can potentially cause the offspring to no longer sample partition H. Each position in a schema can be 0, 1 or *; if I_x is the index of the position of the rightmost occurrence of either a 0 or 1 and I_y is the index of the leftmost occurrence of either a 0 or 1, then the defining length, $\Delta(H)$, is merely $I_x - I_y$.

In the schema theorem, *gains* are ignored and a lower bound is computed for expected *losses.* Both of these actions result in the schema theorem being an inequality which is also a lower bound on the *expected* representation of hyperplane H in the next generation assuming that the population is infinitely large. (Note that it is *not* a lower bound, however,

on the actual representation of H in any finite population.) The *loss* term is bounded as follows: it is assumed that all recombinations between a string contained in H and a string not contained in H will result in a string that does not sample H if crossover falls in the critical part of the schema that corrsponds to H. By choosing the population distribution in an appropriate manner, one can create a worst case situation where the expected gains are zero and expected losses are exactly those predicted by the lower bound on the expected representation given in this version of the schema theorem.

The Goldberg model for strings of length 2 will be used to illustrate the relationship between the various other models. Goldberg noted that the representation of the string 00 in the population of a simple genetic algorithm can be calculated as follows:

$$P_{00}^{t+1} = P_{00}^{t} \frac{f_{00}}{\bar{f}^t}(1 - R'_c \frac{f_{11}}{\bar{f}^t} P_{11}^{t}) + R'_c \frac{f_{01}}{\bar{f}^t} P_{01}^{t} \frac{f_{10}}{\bar{f}^t} P_{10}^{t} \tag{3}$$

where P_j^t is the expected value of the proportional representation of string j in the population at time t, f_j is the evaluation of string j and $\bar{f}_t$ is the average fitness of the population at time t. The variable t is incremented each generation. Goldberg defines R'_c as $R_c\,(\Delta(H)/L - 1)$ in keeping with the spirit of the schema theorem. However, for bit strings of length 2 $(\Delta(j)/L - 1) = 1$, and hence $R'_c = R_c$. The notion of critical crossover regions (Whitley, 1994) replaces the notion of defining length when calculating *losses* for bit strings of length greater than 2. Whitley (1994) shows how to calculate the general proability of *losses* and *gains* for strings of arbitrary length.

Note that this expression can be generalized to compute the representation of the population at time $t+1$ for $P_{01}^{t+1}, P_{10}^{t+1}$ and P_{11}^{t+1} as follows:

$$P_j^{t+1} = P_{00\oplus j}^{t} \frac{f_{00\oplus j}}{\bar{f}^t}(1 - R'_c \frac{f_{11\oplus j}}{\bar{f}^t} P_{11\oplus j}^{t}) + R'_c \frac{f_{01\oplus j}}{\bar{f}^t} P_{01\oplus j}^{t} \frac{f_{10\oplus j}}{\bar{f}^t} P_{10\oplus j}^{t} \tag{4}$$

where $\oplus$ is bitwise exclusive-or over binary strings. Note that any population can be described by the vector $p^t =< p_0^t, p_1^t, p_2^t, p_3^t >$, where p_i^t is the proportional representation of string i in the population at generation t. The index i is obtained by converting the binary representation of a string to a numeric value, so that i refers to string i and also indexes p.

Vose and Liepins (1991) and Whitley et al. (1992) introduced exact models of the simple genetic algorithm using infinite population assumptions. The following notation is based on Vose and Liepins. As noted, the vector $p^t \,\epsilon\, \Re$ is defined such that the k th component of the vector is equal to the proportional representation of string k at generation t. The vector $s^t \,\epsilon\, \Re$ represents the t th generation of the genetic algorithm and the i th component of s^t is the probability that the string represented by i is selected for the gene pool. Thus, s_0^t represents the proportional representation of string 0 in the population at generation t *after* selection has occurred, but before recombination is applied. Likewise, p_0^t represents the proportional representation of string (i.e., the string of all zeros) at generation t *before* selection occurs. Finally, let $r_{i,j}(k)$ be the probability that string k results from the recombination of strings i and j. Now, using $\mathcal{E}$ to denote expectation,

$$\mathcal{E}\; p_k^{t+1} = \sum_{i,j} \; s_i^t \; s_j^t \; r_{i,j}(k) \tag{5}$$

To further generalize this model, a mixing matrix M is constructed where the i,jth entry $m_{i,j} = r_{i,j}(0)$. Here M is built by assuming that each recombination generates a single offspring. The calculation of the change in represention for string $k = 0$ is now given by

$$\mathcal{E}\ p_0^{t+1} \quad = \quad \sum_{i,j} s_i^t s_j^t r_{i,j}(0) \quad = \quad s^T M s \tag{6}$$

where T denotes transpose. Note that this computation gives the expected representation of a single string, 0, in the next genetic population. Vose and Liepins also formalize the notion that bitwise exclusive-or can be used to access various probabilities from the recombination function r. Specifically,

$$r_{i,j}(k) = r_{i,j}(k \oplus 0) = r_{i \oplus k, j \oplus k}(0). \tag{7}$$

This implies that the mixing matrix M, which was defined such that entry $m_{i,j} = r_{i,j}(0)$, can provide mixing information for any string k just by changing how M is accessed. By reorganizing the components of the vector s the mixing matrix M can yield information about the probability $r_{i,j}(k)$. A permutation function, σ, is defined as follows:

$$\sigma_j < s_0, ..., s_{N-1} >^T = < s_{j \oplus 0}, ..., s_{j \oplus (N-1)} >^T \tag{8}$$

where the vectors are treated as columns and N is the size of the search space. The computation

$$(\sigma_q\ s^t)^T M (\sigma_q\ s^t) = p_q^{t+1} \tag{9}$$

thus reorganizes s with respect to string q and produces the expected representation of string q at generation $t+1$. A general operator $\mathcal{M}$ can now be defined over s which remaps $s^T M s$ to cover all strings in the search space.

$$\mathcal{M}(s) = < (\sigma_0\ s)^T M (\sigma_0\ s), ..., (\sigma_{N-1}\ s)^T M (\sigma_{N-1}\ s) >^T \tag{10}$$

Recall that s carries fitness information such that it corresponds to the intermediate phase of the population (after selection, but before recombination) as the genetic algorithm goes from generation t to $t+1$. Thus, to complete the cycle and reach a point at which the Vose and Liepins models can be executed in an iterative fashion, fitness information can now be introduced. A fitness matrix F is defined such that fitness information is stored along the diagonal; the i,i th element is given by $f(i)$ where f is the fitness function. Following Vose and Wright (1993)

$$s^{t+1} = F p^{t+1} / 1^T F p^{t+1} \tag{11}$$

since $F p^{t+1} =< f_0 p_0^{t+1}, f_1 p_1^{t+1}, \ldots, f_{n-1}, p_{n-1}^{t+1} >$ and the population average is given by $1^T F p^{t+1} = f_0 p_0^{t+1} + f_1 p_1^{t+1} + \ldots + f_{n-1}, p_{n-1}^{t+1}$.

Whitley (1993) independently derived equations that can be used to construct the mixing matrix M assuming no mutation occurs and shows how these models relate to earlier work

by Bridges and Goldberg (1987). Nix and Vose (1992; Vose 1993) extend the exact model of the genetic algorithm to look at finite populations using Markov chains.

The Goldberg 2-bit equations, the Bridges and Goldberg model and the executable equations introduced by Whitley all assume that recombination results in 2 offspring, while the Vose and Liepins model assumes crossover results in 1 offspring. An alternative form of M denoted M' can be defined by having only a single entry for each string pair i, j where $i \neq j$. This is done by doubling the value of the enties in the lower triangle and setting the entries in the upper triangle of the matrix to 0. Since each component of s is given by $s_i^t = P_i^t f_i / \bar{f}^t$; this has the rhetorical advantage that

$$(s^t)^T M'(:,1) s_0^t = P_i^t f_i / \bar{f}^t (1 - losses) \tag{12}$$

where $M'(:,1)$ is the first column of M' and s_0 is the first component of s. This isolates a subcomponent of the calculation that is closer to the schema theorem, since it specifically does not include *gains*. Not including the above subcomputation, the remainder of the computation of $s^T M' s$ exclusively calculates string gains. This version of the matrix, M', represents the case where each pair of strings recombines and produces 2 offspring, both of which are retained. In expectation, producing 1 or 2 offspring does not change the exact model, since in any infinite population model, strings must reproduce twice as often when they produce only 1 offspring; on average, they will randomly produce 1 of the 2 possible offspring half the time, and the other possible offspring in the remaining cases. Thus, in expectation in an infinitely large population, the offspring produced are the same.

We have also not considered the effects of probabilistic crossover, which are included in the Goldberg equations and also in the Whitley executable equations. For the Vose and Liepins model, crossover can be included directly in the mixing matrix. The matrix M' is used here in place of M.

2.2 Illustrating Model Equivalence for 2 Bit Strings

We showed by illustration that the Goldberg 2-bit equations are equivalent to the Vose and Liepins model assuming s is scaled by the population average at each generation, so that $s_i^t = P(i,t)(f(i)/\bar{f}^t)$. First, note that

$$P_{00}^{t+1} = P_{00}^t \frac{f_{00}}{\bar{f}^t}(1 - R'_c \frac{f_{11}}{\bar{f}^t} P_{11}^t) + R'_c \frac{f_{01}}{\bar{f}^t} P_{01}^t \frac{f_{10}}{\bar{f}^t} P_{10}^t \tag{13}$$

can be rewritten

$$p_0^{t+1} = s_0^t (1 - R_c s_3^t) + R_c s_1^t s_2^t. \tag{14}$$

The Vose and Liepins model yields the following computation when $R_c = 1$.

$$p_0^t = [s_0, s_1, s_2, s_3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{15}$$

$$p_0^t + 1 = s_0^t(s_0^t + s_1^t + s_2^t) + s_1^t s_2^t \tag{16}$$

and since the componets of the s vector sum to 1,

$$p_0^t + 1 = s_0^t(1 - s_3^t) + s_1^t s_2^t \tag{17}$$

which is equivalent to the results of the Goldberg model when $R_c = 1$. We now show for the 2-bit case that

$$p_0^{t+1} = (s^t)^T M s^t = s_0^t(1 - R_c s_3^t) + R_c s_1^t s_2^t \tag{18}$$

when the probability of crossover is included in the mixing matrix. When the mixing matrix is expressed in lower triangle form, each term in the first column (which expresses string losses) is given by $(1 - R_c + R_c * r_{0,j}(0))$. The matrix entry in all other positions is given by $(R_c * r_{i,j}(0))$, since $i \neq 0$ cannot produce the string 0 when recombination does not occur. When recombination does not occur and $i = 0$, then $(1 - R_c)$ of the 0 strings survive; when recombination does occur and $i = 0$, the 0 strings result with probability $R_c * r_{0,j}(0)$.

We now compute $s^T M s$ accordingly for the 2-bit case, which yields the following result. The values for $r_{i,j}$ can be obtained from the mixing matrix in equation 15.

$$\begin{aligned} p_0^{t+1} = (s^t)^T M s^t \; &= s_0^t \quad \{ \quad (1 - R_c + R_c * r_{0,0})s_0^t + \\ &\qquad\quad (1 - R_c + R_c * r_{0,1})s_1^t + \\ &\qquad\quad (1 - R_c + R_c * r_{0,2})s_2^t + \\ &\qquad\quad (1 - R_c + R_c * r_{0,3})s_3^t\} + R_c s_1^t s_2^t \end{aligned} \tag{19}$$

$$\begin{aligned} &= s_0^t \quad \{ \quad (1 - R_c + R_c)s_0^t + \\ &\qquad\quad (1 - R_c + R_c)s_1^t + \\ &\qquad\quad (1 - R_c + R_c)s_2^t + \\ &\qquad\quad (1 - R_c)s_3^t \quad \} + R_c s_1^t s_2^t \end{aligned} \tag{20}$$

$$\begin{aligned} &= s_0^t \quad \{ \quad (1 - R_c)s_0^t + R_c\ s_0^t + \\ &\qquad\quad (1 - R_c)s_1^t + R_c\ s_1^t + \\ &\qquad\quad (1 - R_c)s_2^t + R_c\ s_2^t + \\ &\qquad\quad (1 - R_c)s_3^t \quad \} + R_c s_1^t s_2^t \end{aligned} \tag{21}$$

$$\begin{aligned} &= s_0^t \quad \{ \quad (1 - R_c)(s_0^t + s_1^t + s_2^t + s_3^t) + \\ &\qquad\quad R_c(s_0^t + s_1^t + s_2^t)\} + R_c s_1^t s_2^t \end{aligned} \tag{22}$$

Since the components of s sum to 1, the above expression yields the following:

$$p_0^{t+1} = s_0^t\{1 - R_c + R_c(s_0^t + s_1^t + s_2^t)\} + R_c s_1^t s_2^t \tag{23}$$

which simplifies to

$$p_0^{t+1} = (s^t)^T M s^t = s_0^t(1 - R_c s_3^t) + R_c s_1^t s_2^t. \tag{24}$$

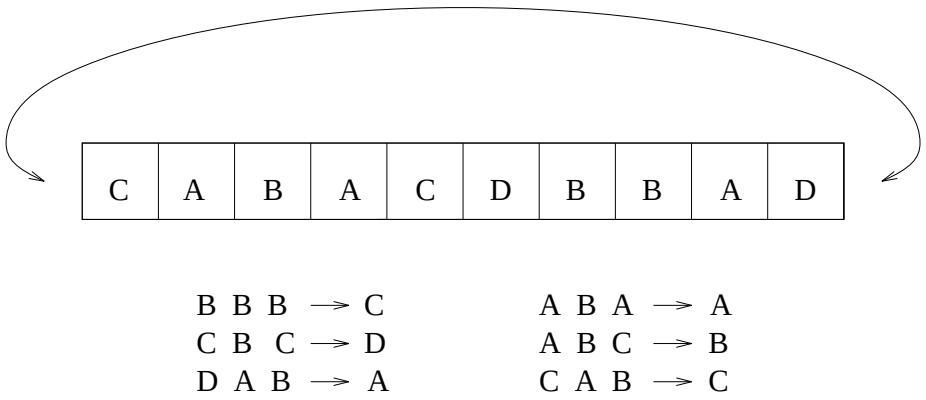

Figure 2: *A simple 1-dimension celluar automata with some examples of rewrite rules.*

3 Cellular Genetic Algorithms

"Massively Parallel" genetic algorithms typically have individual strings residing at cells (processors) that are arranged to create some kind of local structure. For example, strings may be mapped onto a grid. Strings then select mates and recombine with other strings in their immediate neighborhood, which makes this form of genetic algorithm a subclass of cellular automata. Thus, the term *Cellular Genetic Algorithm* has been proposed to describe these parallel algorithms (Whitley, 1993). Manderick and Spiessens (1989) and Hillis (1990) offer two of the first descriptions of such as algorithm. Collins and Jefferson (1991) also use this model and Davidor (1991) specifically calls it the ECO genetic algorithm.

The cellular genetic algorithm uses probabilistic rewrite rules. The cardinality of the symbol alphabet is equal to the number of points in the search space. For now, we consider a 1-dimensional cellular automata which resides on a 1xK array, where K is the population size (See Figure 2). Initially assume selection is omitted and a cell randomly recombines with one of its two neighbors (i.e., neighborhood radius R = 1). Assume a 4 valued alphabet is used: A = 00, B = 01, C = 10 and D = 11. Thus, each rewrite rule has 4 symbols and there are 4^4 possible rewrite rules. Crossover occurs in each cell with probability = 1.0. Two offspring are produced and 1 is randomly chosen as the new string (symbol) in that cell location. The rewrite rules represent the possible outcomes of recombination. Under random conditions there is a 1/4 chance of producing and accepting any one of the 4 possible offspring. A sample of rewrite rules follows.

AAA ⇒ A (P = 1.00)	ACA ⇒ A (P = 0.50)	BCB ⇒ A (P = 0.50)
AAB ⇒ A (P = 0.75)	ACB ⇒ A (P = 0.50)	BCC ⇒ A (P = 0.25)
AAC ⇒ A (P = 0.75)	ACC ⇒ A (P = 0.25)	BCD ⇒ A (P = 0.25)
AAD ⇒ A (P = 0.50)	ACD ⇒ A (P = 0.25)	CAC ⇒ A (P = 0.50)
ABA ⇒ A (P = 0.50)	BAB ⇒ A (P = 0.50)	CAD ⇒ A (P = 0.25)
ABB ⇒ A (P = 0.25)	BAC ⇒ A (P = 0.50)	CBC ⇒ A (P = 0.50)
ABC ⇒ A (P = 0.50)	BAD ⇒ A (P = 0.25)	CBD ⇒ A (P = 0.25)
ABD ⇒ A (P = 0.25)	BBC ⇒ A (P = 0.25)	

The set of rules presented above assumes no mutation. P indicates the probability of writing the symbol A to the center cell. While there are 256 possible rules for this cellular automaton, the above 23 rules are sufficient to construct a complete model. First, these rules are expressed in terms of producing the symbol A. The exclusive-or transformation can again be used to rewrite these rules to cover the cases where other strings (i.e., symbols) are generated. Second, all lefthand sides of rules that are symmetric represent equivalent rules. Thus, the probabilities for rule BBA $\Rightarrow$ A is the same as for ABB $\Rightarrow$ A.

The rules that describe the computational behavior of the cellular automaton can be extracted from the Vose and Liepins (1991) mixing matrix M (c.f. Whitley, 1993) which is constructed by assuming a single offspring is produced during recombination. Let $P_{h,i,j,k}$ be the probability of producing k in the cell occupied by i given its neighbors are h and j. Given a choice of mates h and j, let $\mathcal{S}_{h,j}$ be the probability of selecting h as a mate. Let $\oplus$ denote bit-wise exclusive-or of the binary indices.

$$P_{h,i,j,k} = \mathcal{S}_{h,j}\ m_{(h\oplus k, i\oplus k)} + (1 - \mathcal{S}_{h,j})\ m_{(j\oplus k, i\oplus k)}. \tag{25}$$

This construction can be generalized to 2-dimensional automata corresponding to a cellular genetic algorithm residing on a grid. If cells can reproduce with neighbors to the top and bottom as well as the left and right, then the rewrite rules will be composed of 6 symbols and the selection matrix will have 4 indices, and four terms must be constructed to obtain the desired probability. The computation is simplified when $\mathcal{S}$ or M is sparse.

Given the nature of this model of the cellular genetic algorithm, the probability distribution for a specific string occupying a single cell is the same as the expected distribution of strings in an infinitely large population. Thus, this model does not help us to understand the effects of locality on this particular form of parallel genetic algorithm. To understand the impact of local selection and mating one must use Markov models based on finite population assumptions (e.g., Vose, 1993). However, by viewing the *cellular genetic algorithm* as a type of cellular automaton it may be possible to leverage analytical tools from the field of cellular automata to better understand parallel genetic algorithms.

3.1 Generalizations of the Cellular Genetic Algorithm

One can easily generalize the above model to also cover other forms of genetic search that are much closer to a Simple Genetic Algorithm. First the neighborhood where recombination occurs can be expanded. Simple expansion of the selection neigborhood allows one to model selection schemes that use random walks in the local neighborhood for mate selection. If that neighborhood is expanded to cover the entire population, then a variant of the Simple Genetic Algorithm can be modeled where one parent is chosen by fitness proportional selection over the entire population and the second parent resides in the cell undergoing reproduction. The term Cellular Genetic Algorithm is best applied, however, to models that limit recombination to local neighborhoods, since the selection rules described in this paper would become unwieldy as the neighborhood size grows to encompass the entire population.

4 Conclusions

This paper has reviewed some of the models that exist for simple genetic algorithms. Some properties of the models have also been clarified. In addition, a model has been presented for a parallel cellular genetic algorithm.

References

[1] Bridges, C. and Goldberg, D. (1987) An analysis of reproduction and crossover in a binary-coded genetic Algorithm. *Proc. 2nd International Conf. on Genetic Algorithms and Their Applications.* J. Grefenstette, ed. Lawrence Erlbaum.

[2] Collins, R. and Jefferson, D. (1991) Selection in Massively Parallel Genetic Algorithms. *Proc. 4th International Conf. on Genetic Algorithms*, Morgan-Kaufmann, pp 249-256.

[3] Davidor, Y. (1991) A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. *Proc 4th International Conf on Genetic Algorithms*, Morgan-Kaufmann, pp 257-263.

[4] Goldberg, D. (1987) Simple Genetic Algorithms and the Minimal, Deceptive Problem. In, *Genetic Algorithms and Simulated Annealing,* L. Davis, ed., Pitman.

[5] Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley.

[6] Hillis, D., (1990) "Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure." *Physic D* 42:228-234.

[7] Holland, J. (1975) Adaptation In Natural and Artificial Systems. University of Michigan Press.

[8] Manderick, B. and Spiessens, P. (1989) Fine-Grained Parallel Genetic Algorithms. *Proc. 3rd International Conf. on Genetic Algorithms.* J.D. Schaffer, ed. Morgan Kaufmann.

[9] Mühlenbein, H. (1991) Evolution in Time and Space - The Parallel Genetic Algorithm. *Foundations of Genetic Algorithms*, G. Rawlins, ed. Morgan-Kaufmann. pp 316-337.

[10] Nix, A. and Vose, M. (1992) Modeling Genetic Algorithms with Markov Chains. *Annals of Mathematics and Artificial Intelligence.* 5:79-88.

[11] Spears, W. and DeJong, K. (1991) An Analysis of Multi-Point Crossover. *Foundations of Genetic Algorithms*, G. Rawlins, ed. Morgan-Kaufmann.

[12] Vose, M., (1993) Modeling Simple Genetic Algorithms. *Foundations of Genetic Algorithms -2-,* D. Whitley, ed., Morgan Kaufmann.

[13] Vose, M. and Wright, G., (1993) Simple Genetic Algorithms with Linear Fitness. Unpublished Manuscript.

[14] Vose, M. and Liepins, G., (1991) Punctuated Equilibria in Genetic Search. *Complex Systems* 5:31-44.

[15] Whitley, D., (1993) An Executable Model of a Simple Genetic Algorithm. *Foundations of Genetic Algorithms -2-.* D. Whitley, ed. Morgan Kaufmann.

[16] Whitley, D. (1994) A Genetic Algorithm Tutorial. To appear in *Journal of Statistic and Computing.*

[17] Whitley, D. and Starkweather, T., (1990) Genitor II: a Distributed Genetic Algorithm. *Journal Expt. Theor. Artif. Intell.*, 2:189-214

[18] Whitley, D., Das, R., and Crabb, C. (1992) Tracking Primary Hyperplane Competitors During Genetic Search. *Annals of Mathematics and Artificial Intelligence.* 6:367-388.